НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проект

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»
на тему: «Система трекінгу відвідувачів»

Виконала:
студентка IV курсу, групи ІО-62
Наталія МІРЧУК _____

Керівник:
Доцент, к.т.н.,
Артем ВОЛОКИТА _____

Консультант з нормоконтролю:
Професор, д.т.н.,
Валерій Сімоненко _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність - 123 «Комп’ютерна інженерія»

Освітньо-професійна програма «Комп’ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій СТІРЕНКО

“___”_____2020р.

ЗАВДАННЯ
на дипломний проект студента
Мірчук Наталії Павлівни

1. Тема проекту «Система трекінгу відвідувачів»_____

керівник проекту _____ к.т.н доцент Волокита А.М._____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "07" травня 2020 року №1081-с

2. Термін подання студентом проекту
_____

3. Вихідні дані до проекту __ технічна документація,

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Створення програмної системи для вирішення задачі трекінгу відвідувачів у різного роду приміщеннях за допомогою камер спостереження, використовуючи алгоритми

машинного навчання. Опис предметної області, дослідження засобів обробки програмного забезпечення, розробка алгоритму, програмна реалізація та тестування.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

Принципова схема, функціональна схема та структурна схема

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
нормоконтроль	д.т.н., проф. Сімоненко В.П.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів проекту	Примітка
1	*Затвердження теми роботи*	*1.09.2019*	
2	*Вивчення та аналіз завдання*	*2.09.2019-01.03.2020*	
3	*Розробка архітектури та загальної структури програми*	*02.03.2020-01.04.2020*	
4	*Програмна реалізація*	*02.04.2020-20.04.2020*	
5	*Тестування програмного продукту*	*21.04.2020-01.05.2020*	
6	*Оформлення пояснювальної записки*	*02.05.2020-25.05.2020*	
7	*Передзахист*	*26.05.2020*	
8	*Захист*		

Студент _____ Наталія МІРЧУК

Керівник проекту _____ Артем ВОЛОКИТА

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 4665. 02.000 ПЗ	Технічне завдання	3	
3	А1	ДП 4665. 03.000 ТК	Пояснювальна записка	66	
4	А1	ДП 4665. 04.000 ТК	Принципова схема алгоритму	1	
5	А1	ДП 4665. 05.000 ТК	Структурна схема	1	
6	А1	ДП 4665. 06.000 ТК	Структурна схема	1	
7	А4	ДП 4665. 07.000 ТК	Основні частини коду програми	10	

Технічне завдання до дипломного проекту

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку програми для трекінгу відвідувачів.

Область застосування: розробка системи на основі нейронної мережі для автоматичного трекінгу відвідувачів та підрахунку кількості людей на відео.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання розробки системи трекінгу відвідувачів, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи трекінгу відвідувачів на основі відео з камер спостереження.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в Інтернеті за даним питанням, довідники з програмування та алгоритмів.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Можливість вибору вхідного відео
- Покадрова обробка вхідного відео для розпізнавання людей на ньому
- Проведення аналізу для співставлення людей між кадрами
- Підрахунок людей, що входять в приміщення

5.2. Вимоги до програмного забезпечення

- Unix-подібні операційні системи на базі ядра Linux
- Python 3.6.5 і вище

5.3. Вимоги до апаратного забезпечення

- Комп’ютер на базі процесору Intel Core i3 і вище
- Оперативної пам’яті не менше 4 Гбайт

Анотація

В бакалаврській дипломній роботі реалізовано систему трекінгу відвідувачів, яка призначена для використання на підприємствах та у магазинах для контролю їх вхідного та вихідного потоку, підрахунку конверсії магазину, завантаження магазину в певні години та дні тижня та інших економічних статистик, які є важливими для різних типів бізнесу.

Програмне забезпечення дозволяє на основі відео з камери спостереження підраховувати кількість людей, що заходять у приміщення та виходять з нього. Дана система є окремим та самостійним продуктом. Система була створена та реалізована на мові Python 3.7.5, у середовищі розробки Visual Studio Code.

Ключові слова: нейронна мережа, машинне навчання, комп'ютерний зір, трекінг, Python.

Abstract

The bachelor's thesis implements a system of visitor tracking, which is designed for use in enterprises and stores to control the inflow and outflow, calculate the conversion of the store, load the store at certain hours and days of the week and other economic statistics that are important for different types of business.

The software allows you to count the number of people entering and leaving the room based on video from the surveillance camera. This system is a separate and independent product. The system was created and implemented in Python 3.7.5, in the Visual Studio Code development environment.

Keywords: neural network, machine learning, computer vision, tracking, Python.

Пояснювальна записка
До дипломного проекту
На тему: «Система трекінгу відвідувачів»

Київ – 2020 року

ЗМІСТ

СПИСОК ТЕРМІНІВ ТА СКОРОЧЕНЬ

Датасет	Колекція однотипних даних, що застосовується в задачах машинної обробки даних.
Трекінг	Визначення місцеположення об’єктів, що рухаються за допомогою камери
SORT	(англ. Simple Online Realtime Tracking) назва алгоритму для відстеження об’єктів
GPU	(англ. Graphics Processing Unit) Графічний процесор
CPU	(англ. Central Processing Unit) Центральний процесор
ШІ	Штучний інтелект

ВСТУП

В сучасному світі все більше і більше застосовуються різні комп'ютерні автоматизовані системи, без яких важко уявити теперішню реальність. Важливий внесок в розвиток нових технологій вносять різні системи та алгоритми, розроблені на базі машинного навчання та нейронних мереж. Вони полегшують життя, покращують його якість, та деколи навіть рятують його, завдяки тісній співпраці медицини з машинним навчанням.

Один з найбільш популярних напрямків наукових досліджень в галузі машинного навчання це нейронні мережі, в основі яких лежить бажання імітувати нервову систему людини. Мережа може вчитись на пройдених помилках та набиратись досвіду. Це все дещо узагальнено повинно дозволити змоделювати те, як працює людський мозок. Проте нейронна мережа складається не тільки з математичної моделі, а ще з апаратної чи програмної, як зазвичай, реалізації. Після розробки алгоритму та проведення навчання, отримується готова модель з певними вхідними та вихідними параметрами а такаж необхідними типами даних, в залежності від яких, вона може використовуватись в багатьох різних практичних цілях: в задачах прогнозування, класифікації, для розпізнавання образів, та інших.

Особливу увагу хотілось б звернути на системи, що працюють з фото та відео. Завдяки новітнім комп'ютеризованим технологіям, сучасним програмним засобам для цифрової обробки зображень, стали реальними такі речі як: безпілотні автомобілі, розпізнавання обличчя в телефоні, автоматичне виявлення пневмонії чи інших захворювань легень за рентгенівським знімком, на знімках з космосу можна виявляти початок пожару чи повені, за камерами в приміщеннях можна стежити за безпекою, розпізнаючи лиця людей.

Для підприємств та закладів існує багато способів полегшити роботу, зменшити витрати чи покращити безпеку за допомогою комп'ютерних систем. Одним з них може бути трекінг відвідувачів за допомогою камери біля входу. Це допоможе підраховувати кількість людей, що заходить в магазин, для того

щоб отримати потрібну статистику для підприємства: таку як конверсія завантаження магазину в певні години, та ін. Також до цієї системи можна додати розпізнавання обличь з міркувань безпеки, або для ідентифікації постійних відвідувачів чи покупців. Задача трекінгу відвідувачів може бути вирішена за допомогою нейронних мереж та математичних алгоритмів.

В даній роботі буде розглянуто систему для відслідковування відвідувачів та підрахунок людей, що заходять за допомогою камери спостереження. Така система має ряд переваг над традиційними способами підрахунку економічно важливих показників для підприємств, так як:

- Не потребує постійного контролю зі сторони людини, що зменшує навантаження на працівників
- Підрахунок ведеться в постійному режимі, що дозволяє бачити тенденції в статистиках
- Окрім підрахунку статистик, можна додати інший функціонал, який буде корисний для бізнесу, без залучення персоналу.

РОЗДІЛ 1. ЗАГАЛЬНИЙ ОГЛЯД СФЕРИ ТРЕКІНГУ ВІДВІДУВАЧІВ

1.1 Причини використання

В сучасному світі просто необхідно йти нога в ногу з технологіями, та не відставати ні на мить. Однією з таких нових технологій є системи трекінгу та підрахунку відвідувачів для різного виду приміщень. Це стосується всіх, незалежно від того, чи це малий бізнес чи багатонаціональна компанія, школа чи університет, всім потрібно слідкувати за відвідувачами в першу чергу. Це необхідно для забезпечення безпеки вашого приміщення чи університету та для безпеки всіх його відвідувачів: будь то обслуговуючий персонал, студенти, відвідувачі чи підрядники. Система відстеження відвідувачів дозволяє організаціям контролювати, хто та в який саме момент має доступ до їх приміщення. Ця технологія проста у використанні для персоналу, вона створює безпечне робоче середовище і дозволяє упорядкувати адміністративні завдання. Технологія відстеження відвідувачів має додаткову цінність як інтелектуальна функція звітування. Наприклад, команда з персоналу може використовувати систему для відстеження відвідуваності, усуваючи необхідність у графіку роботи персоналу.

Такі дані допомагають відстежувати ефективність торгового персоналу, аналізувати успішність рекламної діяльності та визначати тенденції відвідування для оптимізації кількості персоналу, необхідного в магазинах. Встановивши системи відстежування роздрібного трафіку, можна отримати багато даних для бізнес-рішень, пов’язаних з маркетинговою діяльністю, графіком роботи персоналу та ефективністю продавців, що призводить до підвищення рентабельності та операційної ефективності.

Щоб краще зрозуміти причини використання такої технології, яка змінює процес керування персоналом та відвідувачами, розглянемо декілька переваг, які переконують в важливості впровадження таких систем.

По-перше, це підвищена видимість. Технологія спостереження за відвідувачами забезпечує безпрецедентний рівень розуміння, якого організації ніколи не мали тоді, коли використовувати традиційну паперову систему. Використовуючи систему спостереження за відвідувачами, можна отримати повноцінні необхідні звіти активності. Отримується інформація про всі куточки вашого бізнесу, яку можна застосувати для підвищення ефективності організації та керування. Одержані звіти дають широкий огляд про те, як відбувалось відвідування приміщення протягом вибраного періоду та інші елементи, які захоче клієнт.

По-друге, це покращена безпека. Це також є важливим пунктом, який, без сумніву, є найбільшою вигодою для багатьох підприємств. Ця вигода є потрійною, оскільки покращує безпеку ваших людей, ваших активів та ваших даних. Це підвищує безпеку людей, оскільки при необхідності можна автоматично викликати необхідні служби за лічені секунди. У разі надзвичайної ситуації це полегшить вам реалізацію евакуації. Крім того, це зменшує загрозу крадіжки в вашому приміщенні, адже часто комерційна крадіжка відбувається тоді, коли люди мають доступ до інформації про приміщення, наприклад, після власного відвідування відповідного магазину чи ін. Нарешті, безпека даних є ключовою причиною, чому підприємства використовують систему відстеження відвідувачів. Таким чином вся необхідна конфіденційна інформація зберігається тільки в одній програмі, а не в кількох місцях, а забезпечити безпеку одного файлу легше ніж багатьох.

Це також дає кращий контроль. Інтелектуальна система відстеження відвідувачів дозволить авторизованим користувачам мати повний контроль над тим, хто та в які моменти має доступ до певних ділянок будівлі. Система може підлаштовуватись під ваші конкретні потреби, питання буде тільки в часі розробки та кількості використаних камер. Наприклад, працівники можуть мати свій значок, який ідентифікує його як не звичайного відвідувача. А оскільки значки посвідчення відвідувача можуть автоматично виводитись на термінал,

кожен відвідувач має певний id, за яким його можна буде розпізнати та визначити коли він у приміщенні, та коли він з нього піде.

Не можна забувати про економію ресурсів. За допомогою системи відстеження відвідувачів ви можете заощадити на ресурсах реєстрації та адміністрації. Для бізнесу чи школи це означає усунути постійні адміністративні витрати на книги для відвідувачів та аркуші для входу. Співробітники приймальні максимізують час, який вони витрачають на професійні завдання, оскільки їм вже не потрібно постійно керувати відвідувачами та доставками. Для керованої офісної будівлі чи бізнес-центру ця технологія надає можливість перейти на електронний прийомом. Без потреби додаткового працевлаштування нового персоналу для прийому, це може бути досить великою економією.

Ще одним із важливих позитивних аспектів трекінгу відвідувачів є отримання економічно важливих показників, таких як конверсія, загруженість приміщення по годинах, середня кількість відвідувачів.

В маркетингу під конверсією, як правило, розуміється частка візитів у ваш магазин (або інше комерційне приміщення), в ході яких відвідувачі вчинили цільову дію (зазвичай покупку продукту або послуги).

Під цільовою дією може матися на увазі покупка товару в магазині, оплата курсу сеансів масажу, запис на стрижку чи курс, запис в книзі відгуків, реакція на рекламу, отримання візитної карти, та інші. На кожному етапі воронки продажів конверсія може бути різною і по-різному впливати на подальші стратегії розвитку.

Наприклад, певну рекламну вивіску побачили 1000 людей. Після цього в магазин зайшло 600 чоловік. Конверсія може виступати в ролі відносини показів даної реклами (кількості людей, що отримали візитку, кількості людей, що пройшли повз вивіску) до кількості відвідувань магазину (чи іншого приміщення). В даному випадку, якщо ставите метою саме прихід відвідувача, конверсія складе 60%. Для того, щоб збільшити конверсію на даному етапі

воронки продажів, необхідно створити якісне, релевантне оголошення, яке буде привертати увагу та затримувати погляд людей. [1]

Завантаженість комерційних приміщень по днях\годинах – можливість правильно регулювати кількість працівників та їхню загрузку. А також дає інформацію щодо вигідних годин в магазині (іншому комерційному приміщенні), а потім і регулювати тривалість, та час дії знижок, промо та акцій.

Системи трекінгу можуть бути реалізовані кількома способами:

- Аналіз потокового відео з камер спостереження та його обробка за допомогою нейронних мереж по розпізнаванню образів людини та різних математичних алгоритмів.
- Обробка відео за допомогою нейронних мереж по розпізнаванню певних контрольних точок (бейджів) та різних алгоритмів.
- Установка датчиків руку на кожних дверях та контроль за допомогою отриманої інформації

Поряд із вищезазначеними причинами, ця технологія пропонує велику віддачу від інвестицій та максимізує інвестиції, вже витрачені на системи доступу.

1.2 Приклади комерційних та некомерційних приміщень для використання трекінгу відвідувачів

Роздрібна торгівля. Аналітика роздрібної торгівлі – це процес збору аналітичних даних із програмного забезпечення вашої роздрібної торгівлі, щоб забезпечити краще розуміння своїх клієнтів та їхньої поведінки та в кінцевому підсумку покращити ефективність роботи.

Використання систем відстежування клієнтів може допомогти вашому роздрібному магазину оптимізувати маркетингові та операційні стратегії, за допомогою збору необхідної інформації. Цілком доступні дані роздрібної

аналітики роздрібної торгівлі дозволяють роздрібним торговцям організовувати макети своїх магазинів, вітрини та товари відповідно до їхніх характеристик та потреб. Лічильники відвідування роздрібних магазинів мають вирішальне значення для управління розподілом персоналу для максимального задоволення клієнтів та оптимізації ресурсів.

Бібліотеки та музеї. Це неприбуткові організації, тому вони не приносять великого доходу. Ці типи місць покладаються на державне фінансування, тому вони повинні виправдовувати свої потреби у фінансуванні та представляти статистичні дані, щоб мати можливість продовжувати своє існування. Тут люди потрібні для того, щоб складати точні звіти про кількість відвідувачів/меценатів, які щодня, щотижня, щомісяця або щорічно відвідують бібліотеки чи музеї.

Аеропорти приймають найбільш динамічний трафік у світі. Відстеження руху натовпу з моменту проходження перевірки безпеки до того, як вони потраплять у літак, є вирішальним для ефективних операцій, розподілу персоналу та позитивного досвіду. Зазвичай пасажири бажають витрачати менше часу на очікування біля прилавків або перевірки безпеки. Натомість вони хочуть скористатися можливістю робити покупки без податкових зобов'язань. І в цьому випадку застосування систем відстежування відвідувачів буде дуже корисним.

Торгові центри. Торгові центри приваблюють величезну кількість відвідувачів; здебільшого через широкі можливості для покупок, які вони надають, а також заради розважальної цінності.

Менеджери торгових центрів можуть формувати траєкторію руху кожного відвідувача, якщо вони мають необхідний досвід та навички для професійного маніпулювання. Крім того, вони можуть досягти оптимальної ефективності у своїх торгових центрах, вимірюючи трафік відвідувачів у кожному приміщенні, регулюючи оренду цих приміщень, відповідно, і

виправдовуючи цінність своїх орендарів, використовуючи підрахунок людей у торгових центрах [2].

1.3 Огляд готових рішень на ринку

1.3.1 V-count

Однією з компаній, що пропонує рішення трекінгової системи на ринку це V-count.

Останній їхній продукт забезпечує середню 98% точність підрахунку точності за допомогою своєї технології стерео зору, надійної технології, доступної зараз на ринку людей, що працюють в цій сфері. Це один з найточніших та надійніших спосіб вимірювання активності людей, та знаходження їхньої траєкторії.

Функціонал даної системи забезпечує одночасний підрахунок входу та виходу людей та повідомлення про ці цифри окремо. Крім того, віднімаючи вихід із введення рахунку, ви можете мати уявлення про те, коли в приміщенні знаходиться найбільша кількість людей, а отже знати години піку для відвідувачів.

V-Count призначений для роздрібних торгових мереж, розважальних закладів, операторів торгових центрів, закладів охорони здоров’я та інших, хто хоче зрозуміти, як і в яких кількостях люди входять, пересуваються і виходять з фізичних приміщень. Підрозділ може забезпечити широкий діапазон показників трафіку, черги та інших поведінкових показників, надаючи додаткову інформацію про ваших клієнтів та їх поведінку в магазинах.

Переваги цього продукту:

- Підрахунок людей ведеться у реальному часі. Можна підрахувати кількість людей, які в реальному часі входять, виходять та проходять повз ваш магазин чи інше приміщення.
- Конверсія. Можна обчислювати коефіцієнти конверсії кожного магазину та кожного входу, і таким чином знати, яка частина з ваших клієнтів забезпечують ваші продажі.

- Порівняння ефективності магазинів в мережі. Можна визначати високоефективні магазини та оптимізувати свої найменш прибуткові магазини, щоб досягти їхнього росту.
- Пікові години. Продукт дає можливість дізнаватись, коли ваші магазини генерують найбільшу кількість продаж та відвідувань.
- Оптимізація персоналу Отримуючи дані з цього продукту можна вдало оптимізувати роботу власного персоналу, відстежуючи кількість відвідувачів та їхні потреби в магазинах протягом годин роботи.
- Груповий підрахунок. Зробіть свій коефіцієнт конверсії більш точним, консолідувавши групи відвідувачів, сімей та пар як одного потенційного покупця. [3]

1.3.2 SensMax

Ще однією популярною системою є SensMax – програма для підрахунку людей у роздрібних підприємствах, яка допомагає відстежувати кількість потенційних клієнтів, що відвідують ваші магазини. Дана система точно відстежує статистику відвідувачів і поєднує її з фінансовими даними з кас. Це допомагає визначити, який дохід від продаж генерується одним відвідувачем та яка середня кількість товарів приходиться на одного відвідувача.

Система підрахунку клієнтів SensMax, призначена для збору статистики роздрібної торгівлі в мережі роздрібних магазинів. Датчики, які рахують людей, - це бездротові пристрої, тому їх легко встановити без жодних кабельних робіт.

Система підрахунку клієнтів складається з трьох компонентів – бездротових датчиків лічильника дверей, шлюзу даних для збору даних з лічильників руху дверей та програмного забезпечення для звітування, що відображає дані з лічильників руху дверей у різних звітах.

Існує можливість встановити програмне забезпечення так, щоб звітування відбувалось на ваші сервери або використовувало хмарний сервіс звітності SensMax для обробки даних. Хмарний додаток для звітування зберігає

дані на надійних серверах, тому для обробки статистичних даних вам не потрібно створювати локальну IT-інфраструктуру [4].

Програмне забезпечення для звітування дозволяє перевіряти статистику відвідувачів клієнтів, тенденції відвідувань, порівнювати періоди до і після маркетингових заходів або навчання персоналу з продажу, перевіряти коефіцієнти конверсій, час відкриття та закриття магазинів .

1.3.3 Linkanalytix

Останньою розглянутою системою буде розробка Linkanalytix, яка обіцяє досягнення більш, ніж 10% збільшення коефіцієнта конверсії.

Мета підрахунку кількості людей - це не сам підрахунок, найголовнішим є те, що ви можете зробити з результатами підрахунку. Якість результатів залежить від програмного та апаратного забезпечення продукту. Підрахунок повинен забезпечувати точні та стабільні числа, і повинен бути надійним. Справжнім ключем до успіху в роздрібній торгівлі є спосіб використання цифр для покращення результатів. За допомогою подібного програмного забезпечення можна зробити такі важливі речі:

- Збільшити трафік
- Збільшити продажі
- Збільшити конверсію

Система відстежування людей є одними з найновіших гаджетів AI, спроектованих для спрощення отримання показників у роздрібній торгівлі та покращення їхньої якості. Одним з найкращих показників, який можна отримати для вашого бізнесу, є покупець. Такі системи дають інформацію про години, коли люди найбільше відвідують ваш магазин, ті частини вашого магазину, де вони відвідують найбільше, а також запобігають крадіжці

Підвищення коефіцієнта конверсії, обсягу продажу та прибутковості є головними цілями кожного бізнесу. За допомогою системи підрахунку відвідувачів можна проводити оптимізацію в магазині, дізнаючись кількість відвідувачів, щодня, щодня, щотижня або сезону. Також важливим є

оптимізація багатьох важливих факторів, таких як планування співробітників, розподіл ресурсів та зберігання звітів із даними в режимі реального часу.

Найголовнішим для вашого бізнесу – це максимізація прибутків, яка напряму залежить від ваших клієнтів. Від того, чи вони задоволені якістю роботи персоналу, чи в магазині є достатня кількість необхідного товару, чи достатньо безпечно знаходитись в магазині. Система підрахунку людей LinkAnalytix забезпечує високу точність та зручні лічильники для бізнесу.

Загалом кожна з наведених систем має свої переваги, недоліки та особливості, які відрізняють її від інших наявних на ринку (у таблиці 1.1).

Таблиця 1.1.

Система	V-count	SensMax	LinkAnalytix
Спосіб трекінгу	Стерео-камери	Датчики руху на дверях	Звичайні камери спостереження
Можливість підрахунку людей	+	+ (проте дуже неточна, через застарілий спосіб)	+
Швидкість роботи	В реальному часі	В реальному часі	Близько 1.5 години на 1 годину відео

Як видно з таблиці, існуючі системи дозволяють виконувати трекінг відвідувачів за допомогою різних способів: перша система за допомогою стерео-камер, які є доволі дорогими, друга за допомогою датчиків руху, що робить обчислення кількості людей дуже неточним, а третя за допомогою звичайних камер спостереження, що є оптимальним вибором. Таким чином, актуальною, на мою думку, є подальша розробка системи на основі звичайних камер спостереження, а основним покращенням повинна бути швидкість роботи, що визначається якістю та новизною використаних алгоритмів.

ВИСНОВКИ ДО РОЗДІЛУ 1

В сучасному світі важко обійтись без постійного аналізу оточуючого середовища та його вивчення. Одним із способів аналізу є трекінгові системи відвідувачів різного виду приміщень, які розглянуті в цьому розділі. Їхнє використання спричинено необхідністю в інформації про приміщення, покращенні його безпеки, економії ресурсів та отримання економічно важливих статистичних даних.

Такого виду системи тільки набувають популярність, та мають великий потенціал, оскільки можуть застосовуватись практично в усіх видах публічних приміщень, без різниці комерційні вони чи ні. Таким чином вони знайдуть місце в звичайних магазинах, торгових центрах, бібліотеках, аеропортах та вокзалах.

Попри те, що це порівняно новий вид систем, на ринку можна знайти достатню кількість готових рішень. В даному розділі було проаналізовано три найбільш популярні системи з представлених на сьогоднішній день. Вони являються одними з найкращих на ринку. Кожна з наведених систем має свої переваги та недоліки, а також використовує власні унікальні способи трекінгу, такі як: трекінг за допомогою відеокамер, за допомогою датчиків руху та, наприклад, за допомогою датчиків локації. Вивчення та аналіз існуючих систем є необхідним пунктом для якісної розробки власного програмного забезпечення.

РОЗДІЛ 2. ОГЛЯД МАШИННОГО НАВЧАННЯ

2.1. Традиційне машинне навчання

Алгоритми машинного навчання використовують статистику для пошуку шаблонів у масових обсягах даних. А дані охоплюють багато речей – цифри, слова, зображення, кліки, все, що нас оточує. Якщо вона може бути збережена в цифровому форматі, її можна подати в алгоритм машинного навчання.

Машинне навчання – це процес, що забезпечує багато служб, якими ми користуємось сьогодні – рекомендаційні системи, пошукові системи, канали соціальних медіа; голосові помічники, та інші. У всіх цих випадках кожна платформа збирає якомога більше даних про вас – які жанри ви любите дивитися, на які посилання ви натискаєте, на які статті ви реагуєте – і за допомогою машинного навчання будує здогадки про те, що ви може захотіти наступним.

Традиційне машинне навчання, на базі математичних алгоритмів буде добре справлятись з задачами класифікації, регресії та кластеризації, проте для задач розпізнавання та локалізації об’єкта на фото краще звернутись до глибинного навчання та нейронних мереж.

2.2 Глибинне навчання

Методи глибинного навчання (і алгоритми ШІ в цілому) мають дуже великий потенціал для того щоб впливати на розвиток технологій. Особливо тих, які базуються на аналізі фото та відео, адже цифрові елементи найзручніше аналізувати за допомогою штучних інтелектуальних алгоритмів.

Більшість сучасних моделей глибокого навчання ґрунтуються на штучних нейронних мережах. У процесі глибокого навчання кожен рівень мережі вчиться перетворювати свої вхідні дані у дещо більш абстрактне та

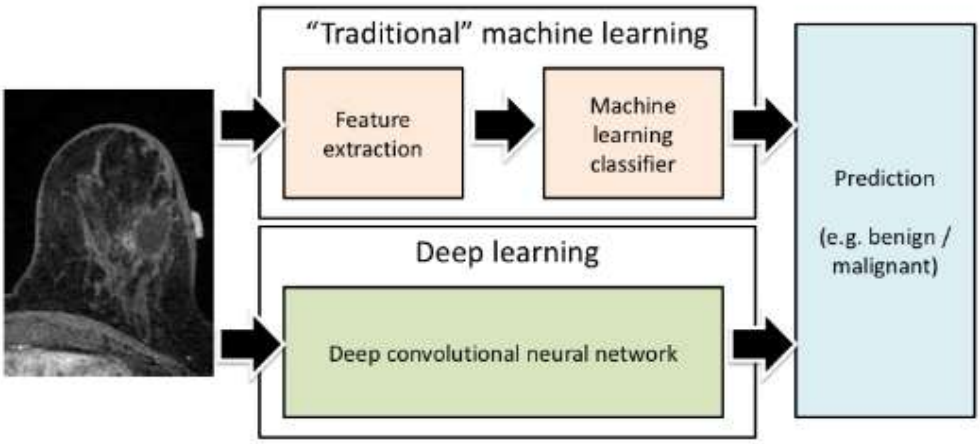
складене. У програмі розпізнавання зображень необроблений вхід може бути матрицею

пікселів; перший репрезентативний шар може абстрагувати пікселі та кодувати краї; другий шар може складати і кодувати розташування ребер та ліній; третій шар може кодувати певні збірні образи; і четвертий шар може визнати, що зображення містить обличчя. Важливо, що в процесі глибокого навчання можна дізнатися, які особливості оптимально розміщувати на якому рівні. (Звичайно, це не повністю усуває необхідність ручної настройки; наприклад, різні кількості шарів та розмірів шарів можуть забезпечити різний ступінь абстрагування.)

Слово "глибокий" у "глибокому навчанні" означає кількість шарів, через які дані перетворюються, Глибокі моделі (з кількістю шарів більше 2) здатні отримати кращі характеристики, ніж дрібні моделі, а отже, додаткові шари допомагають ефективно вивчати функції.

2.3 Глибинне навчання проти «традиційного» машинного навчання

Все частіше ми чуємо про різницю між глибинним навчанням і «традиційним», тобто машинним навчанням (Рис. 1). Різниця є важливою, особливо в плані візуалізації. Для традиційного машинного навчання, стандартним першим кроком є виділення ознак. Це означає, що для обробки певного об'єкта необхідно вирішити, які його характеристики об'єкта будуть значимими і реалізовувати алгоритми на їхній основі. Для вирішення цієї задачі було реалізовано певний набір алгоритмів для вилучення різноманітних характеристик, таких як: вилучення текстури, розмірів, форми та інших. Цей процес значною мірою є довільним, оскільки дослідник або практик, що займається вирішенням проблеми, часто повинні робити припущення, які характеристики будуть корисні для конкретної задачі. Таким чином, ризикуючи включити в алгоритм непотрібні та надлишкові функції.



.

Рис. 2.1. «Традиційне» машинне навчання та глибинне навчання

У процесі глибинного навчання процес прийняття рішень щодо вилучення ознак виконується власне мережею, і окрім того з кожним кроком вона вчиться краще виявляти потрібні характеристики. Однак, для того, щоб нейронна мережа самостійно вибирала необхідні функції та параметри потрібні набагато більші набори даних [5].

2.4 Визначення нейронної мережі

Один з найбільш популярних напрямків наукових досліджень в галузі машинного навчання є нейронні мережі, набір алгоритмів, завданням яких є зімітувати роботу мозку людини для розпізнавання закономірностей в наборах даних. Вони оброблюють отримані дані за допомогою своєрідного машинного сприйняття, позначень або поділу (кластеризації) вхідних даних. Закономірності, розпізнані нейронними мережами, є числовими, які містяться у векторах. За допомогою такого типу представлення інформації, можна перекласти всю інформацію, що нас оточує: зображення, звук, текст або часові ряди.

Нейронні мережі не програмуються в звичному для нас сенсі цього слова, а навчаються. Можливість навчатися – одна з головних переваг нейронних мереж перед традиційними алгоритмами програмування. Технічно навчання полягає в знаходженні коефіцієнтів між елементами мережі.

Протягом навчання нейронна мережа може виявляти складні залежності між вхідними і вихідними даними, а також виконувати певні узагальнення над ними. Мережа може вчитись на пройдених помилках та набиратись «досвіду». Це все, дещо узагальнено, але повинно дозволити математично змоделювати те, як працює людський мозок та його нейронні сполучення.

Нейронні мережі можуть вирішувати різні задачі, такі як кластеризація та класифікація. Вони допомагають згрупувати нерозмічені дані відповідно з їхньої схожістю, і класифікують різні дані, коли вони мають певний набір результуючих даних для навчання. (Мережі також вміють самостійно отримувати певні особливості, які надходять до алгоритмів класифікації та кластеризації, тому можна уявляти глибинні нейронні мережі як складові компоненти більших додатків машинного навчання, що включають алгоритми класифікації та регресії).

Глибоке навчання, по своїй суті, робить відображення вхідних даних мережі на її вихідні дані. Воно знаходить кореляцію або причинний зв'язок між ними і навчається наближати деяку невідому нам функцію f (x) = y між будь-яким входом x і будь-яким виходом y, роблячи припущення, що вони залежні.

Класифікація це одна з найбільш популярних задач, що вирішуються нейронними мережами. Вона повинна отримати на вхід набір даних з певними мітками, які характеризують клас даного набору даних. Тоді класифікатор зможе знайти кореляцію між мітками та даними, для того щоб потім самому визначати необхідну мітку на нових даних.(Рис. 2.) Таку мережу-класифікатор відносять до класу «навчання з учителем». Вона може вирішувати такі задачі:

- Класифікація тексту, наприклад електронних листів, на «спам» і «не спам»
- Розпізнавання настрою тексту (наприклад серед відгуків)
- Оцінювання кредитоспроможності позичальників (бінарне рішення про те, чи надавати клієнту банку кредит на основі загальної інформації про людину)
- Розпізнавання жестів у відео

- Розпізнавання мови, розпізнавання почуття голосу
- Розпізнавання символів та рукописного тексту
- Передбачення відтоку клієнтів

Для навчання нейронної мережі можна використовувати практично будь-які дані, що корелюють з будь-якими мітками.

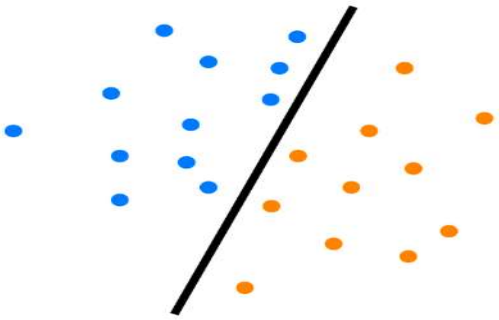


Рис. 2.2. Узагальнене зображення роботи класифікатора[6]

Ще однією задачею нейронних мереж є кластеризація або групування, тобто виявлення подібності. Даний тип задачі не вимагає маркування вхідних даних. Навчання на нерозмічених даних називається «навчання без учителя». Такі дані являють собою, більшу частину даних у всьому світі, адже це є дані, що оточують нас кожного дня. Відомо, що чим більше даних використовується для навчання нейронної мережі, тим точнішим буде її результат. Таким чином, «навчання без вчителя» може генерувати високоточні моделі.

Кластеризація полягає в тому, що мережа сама виявляє певні спільні ознаки в вхідних даних і групує їх в певні «кластери»(Рис. 3.). Дана мережа може вирішувати такі задачі:

- Виділення груп людей на основі графа сполучень в соціальних мережах.
- Підбір рекомендацій для користувача на основі вподобань чи закономірностей інших користувачів в даному кластері.
- Розбиття вибірки на групи схожих об'єктів для спрощення наступної обробки даних і прийняття рішень, застосовуючи до кожного кластеру свій метод аналізу

- Зменшення розміру даних. Якщо вихідна вибірка занадто велика, то можна зменшити її, залишивши по одному, найбільш типовому представникові від кожного кластера.
- Виявлення аномалій, шляхом знаходження нетипових об'єктів з набору даних, які не вдається приєднати до жодного з кластерів.

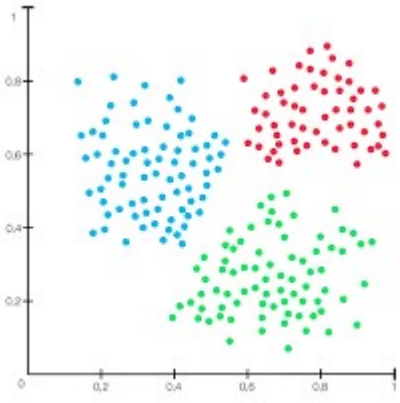


Рис. 2.3. Кластеризація даних [7]

Здатність нейронних мереж до прогнозування напряму залежить від їхніх власних можливостей до узагальнення та виокремлення прихованих і непомітних залежностей між вхідними та вихідними даними. Після такого навчання мережа є здатною передбачити наступне значення певної послідовності даних на основі декількох попередніх значень та існуючих чинників, при умові, що вони наявні (Рис. 4.).

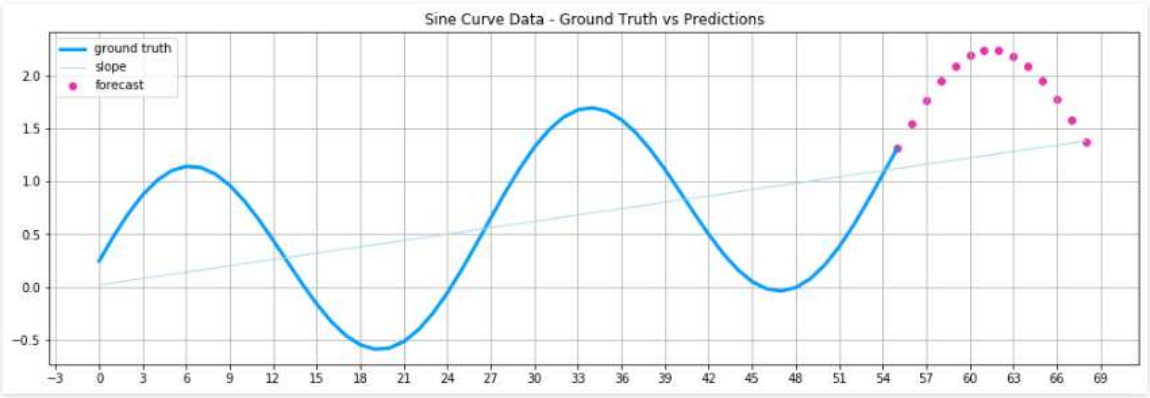


Рис. 2.4. Приклад роботи мережі для прогнозування значень[8]

Варто зазначити, що прогнозування наступних значень є можливим тільки тоді, коли попередні його зміни в якійсь мірі визначають чи формують майбутні. Наприклад, прогнозування вартості акцій на основі їхньої вартості за

минулий тиждень чи місяць скоріш за все виявиться успішним, тоді як прогнозування результатів наступної лотереї навіть на основі даних за останні 50 років майже точно не дасть ніяких результатів. Прогнозування може вирішувати такі задачі:

- Поломки обладнання в центрах обробки даних, на виробництві або в сфері транспорту
- Прогноз фінансових індикаторів
- Передбачення хвороби, наприклад інсульту чи серцевого нападу на основі життєво важливої статистики, отриманої з певного медичного пристрою
- Відтік клієнтів, тобто прогнозування ймовірності втрати клієнта на основі веб-активності та метаданих

Добре відомо, що чим краще можна передбачити майбутню проблему, тим краще можна не дати їй статись. З нейронними мережами можна жити з меншою кількістю непередбачуваних моментів, що дозволяє краще контролювати ситуацію, та планувати подальші дії [9].

Після швидкого огляду варіантів застосування глибинного навчання в різних сферах життя людей та підприємств, варто дізнатись, з яких елементів складаються нейронні мережі .

2.5 Складові елементи нейронної мережі

Раніше було згадано, що нейронні тобто мережі, що складаються з декількох шарів і у процесі глибинного навчання кожен рівень вчиться перетворювати свої вхідні дані у дещо більш абстрактне.

Кожен шар нейронної мережі складається з вузлів, тобто просто з місця, де проводяться необхідні обчислення. Вузол з'єднує вхідні дані з коефіцієнтами або вагами, які в свою чергу або посилюють, або зменшують вплив цього входу, тим самим регулюючи правильну роботу на основі минулих обчислень(минулих коефіцієнтів) (Рис.5). Результат цієї вхідної маси підсумовується, а потім сума передається через функцію активації вузла, щоб

визначити, як цьому сигналу рухатися по мережі. Якщо сигнали проходять, нейрон "активується".

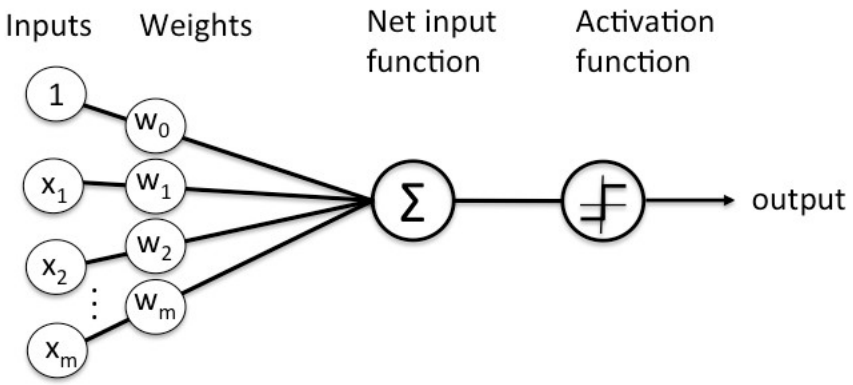


Рис. 2.5. Вузол нейронної мережі

Шар нейронної мережі – це ряд вузлів, які вмикаються або вимикаються, коли вхід подається (або не подається) через мережу в залежності від коефіцієнтів та функції активації. Вихід кожного шару по суті є також входом наступного шару вузлів, починаючи з початкового рівня, який отримує ваші початкові дані (Рис. 6.).Тобто в результаті маємо:

- Вхідний шар приймає вхідні дані і передає їх в перший прихований шар.
- Приховані шари виконують математичні обчислення з вхідними даними. Одне із завдань при створенні нейронних мереж – визначення кількості прихованих шарів і нейронів на кожному шарі. Прихованих шарів може бути декілька.
- Вихідний шар видає кінцевий результат.

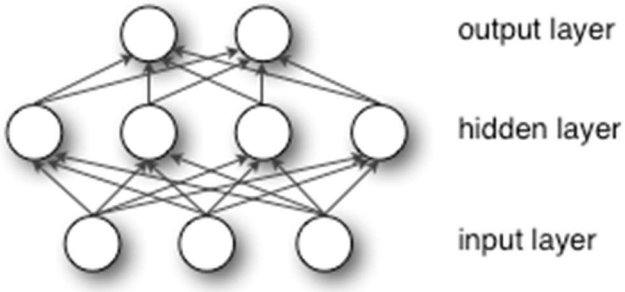


Рис. 2.6. Шари нейронної мережі

2.6 Принципи роботи нейронних мереж

Термін «глибинні нейронні мережі» відрізняються від більш загальноприйнятих мереж тим, що в своєму складі має більш ніж один прихований шар.

Найперші варіанти нейронних мереж, наприклад перші перцептрони, були не були глибинними, так як складалися тільки з одного вхідного і одного вихідного шару, без прихованого шару між ними. Зараз вважається, що коли в мережі більше трьох шарів (включаючи вхід і вихід), тоді це не просто мережа, а «глибинна» мережа. Тоді, беручи це до уваги, «глибинне» це не просто модна приставка до слова, а строго визначений термін.

У мережах глибинного навчання кожен рівень вузлів навчається на певному наборі функцій, які є вихідними даними попереднього рівня. Просуваючись далі в нейронну мережу, наступний шар може розпізнавати більш складні функції, образи та залежності, оскільки вони поєднують і рекомбінують функції попереднього рівня (Рис. 7).

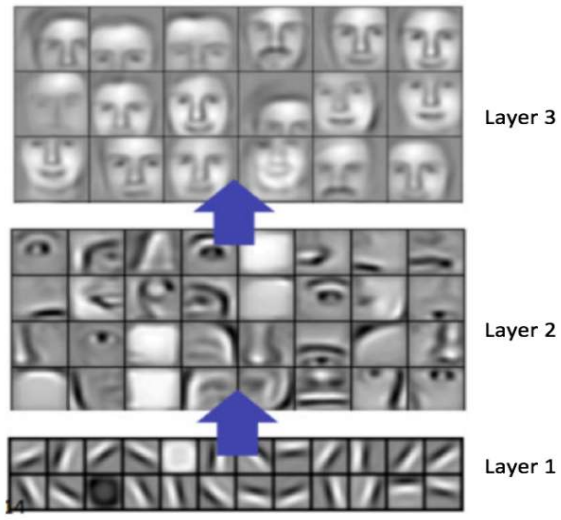


Рис. 2.7. Приблизне зображення ієрархії шарів нейронної мережі

Це так звана, ієрархія функцій, тобто зростаюча складність та абстракція. Завдяки такій ієрархії мережі глибинного навчання мають змогу обробляти великі, багатовимірні набори даних з мільйонами параметрів, що проходять через різні нелінійні функції.

2.7 Мережі "Feedforward"

Вся суть машинного навчання полягає у отриманні максимально точного результату за максимально короткий час. Тобто нам потрібно якнайшвидше навчитись мінімізувати результуючу помилку. Оскільки під час навчання моделі ми проходимо ті ж самі пункти в циклі стільки разів, скільки потрібно. Початком циклу є момент ініціалізації вхідних даних та ваг, а закінченням – отримання фінального результату з мінімальною похибкою.

Набір ваг, незалежно від того, чи вони є у початковому, фінальному чи проміжному стані, також може називатися моделлю, оскільки він є певною спробою моделювання відношення даних до міток, для розуміння структури даних і майбутніх прогнозів.

Це можна пояснити тим, що на початку вагові коефіцієнти нейронної мережі зазвичай ініціалізуються випадково. Невідомо, як їх потрібно змінювати для мінімізації похибки. Тому спершу починаємо з припущення, і методом спроб і помилок рухаємось до кращого результату. Мережа в такому випадку схожа на дитину: вона народжується, нічого не знаючи, не знаючи багато, і потім, завдяки впливу досвіду та власних помилок, вона дізнається про світ.

У мережу приходять певні вхідні дані, ініціалізуються коефіцієнти або ваги, які потім вводяться до множини припущень.

$$input * weight = guess \quad (1)$$

Отриманий результат повинен зробити припущення, що таке вхідні дані. Тоді нейрон порівнює припущення з правильним варіантом і знаходить помилку, просто знаходячи різницю між правдою і отриманим результатом.

$$ground\ truth - guess = error \quad (2)$$

Для подальшого кроку модель повинна «навчитись на своїх помилках» і зробити висновки. Мережа вимірює отриману помилку і повертає її назад по шарах моделі, змінюючи ваги таким чином, щоб мінімізувати помилку.

$$error * weight's\ contribution\ to\ error = adjustment \quad (3)$$

Ці три формули вище описують три основні функції роботи нейронних мереж: опрацювання вхідних даних, отримання результату та знаходження

помилки і, власне навчання, застосування оновлених ваг до моделі - щоб почати це коло знову.

2.8 Множинна лінійна регресія

В основі всіх алгоритмів машинного навчання, включно з штучними нейронними мережами лежить всього лиш запрограмована математика. Тому для розуміння того, як працює глибинне навчання достатньо знати основи статистики, а саме лінійну регресію, яка у найпростішому варіанти виглядає, як:

$$Y_{hat} = bX + a \quad (4)$$

де Y_hat є отриманим результатом, X є входом, b - нахилом і a – перетином з віссю y (Наприклад, можна припустити для конкретизації: X може бути кількістю викурених сигарет, а Y_hat може бути ризиком раку; X може бути кількості віджимання і Y_hat може бути скинутою вагою; X – кількість опадів, а Y_hat ймовірністю втрати урожаю). Таким чином, кожна зміна X викликає за собою зміну Y_hat. Ця проста взаємозалежність між двома змінними є основою лінійної регресії.

Наступним кроком є лінійна регресія з більшою кількістю вхідних даних, які впливають на вихідну змінну. Тобто множинна лінійна регресію. Зазвичай вона виглядає так:

$$Y_{ha} = b_1 * X_1 + b_2 * X_2 + b_3 * X_3 + a \quad (5)$$

(Щодо прикладу з раком, можна додати кількість випитого алкоголю та кількість радіаційного опромінення. В такому випадку всі названі змінні впливають на Y_hat.) [10].

В такому вигляді, множинна лінійна регресія спрацьовує на кожному вузлі кожного шару нейронної мережі. В кожному вузлі одного шару вхідні дані є рекомбінованими, в залежності від відповідних вагових коефіцієнтів, попереднього шару. Значить входи змішуються в різних пропорціях, які відрізняються для кожного вузла з наступних шарів, адже матриця ваг повністю різна. Таким чином, мережа випробовує різні комбінації вхідних даних і коефіцієнтів, яка буде найбільше зменшувати помилку.

Після того, як всі вхідні дані одного вузла, щоб отримати Y_hat, результат передається через певну нелінійну функцію, так звану функцію активації. Перетворення в кожному вузлі зазвичай є S-подібними, подібними до логістичної регресії. Вихідні дані всіх вузлів, потім стискаються в простір, розділений нелінійною функцією між 0 і 1.

2.9 Градієнтний спуск

Градієнтний спуск це популярний ітеративний алгоритм оптимізації для знаходження мінімуму функції, який регулює вагові коефіцієнти в залежності від. викликаної ними помилки.

Також, можна дати таке визначення градієнту – це міра напрямку та швидкості найшвидшого росту функції. Щоб знайти локальний мінімум функції з використанням градієнтного спуску, потрібно зробити кроки, пропорційні негативу градієнта (або приблизного градієнта) функції в поточній точці..

Під час свого навчання, нейронна мережа на кожному кроці налаштовує багато вагових коефіцієнтів, для правильного відображення сигналу. Залежність помилки від ваг є градієнтом, тобто $\frac{dE}{dw}$, яка вимірює ступінь внеску невеликої зміни ваги в відповідну зміну помилки.

У звичній нам мережі взаємозв'язок між ваговим коефіцієнтом по похибкою мережі зображується так:

$$\frac{dError}{dweight} = \frac{dError}{dactivation} * \frac{dactivation}{dweight} \quad (6)$$

По великому рахунку, суть глибинного навчання полягає в тому, щоб правильно та швидко коригувати матрицю вагових коефіцієнтів моделі в залежності від отриманої помилки. Навчання в такому випадку закінчується тоді, коли отримується бажаний рівень похибки (або максимально приближений до нього).

2.10 Функції активації

Раніше було згадано, що функція активації визначає вихід, який буде видавати вузол, беручи за основу його вхід. Це робиться для певної нормалізації сумарного результату. Стандартну інтегральну схему функції активації можна

розглядати як цифрову мережу функцій, що може бути "ON" (1) або "OFF" (0), в залежності від входу. Це схоже на поведінку лінійного перцептрона в базових нейронних мережах. Однак тільки нелінійні функції активації дозволяють мережам вирішувати нетривіальні проблеми, використовуючи лише невелику кількість вузлів.

2.11 Логістична регресія

Попри те, що нейронні мережі, навчаючись на розмічених даних, повертають двійковий вихід, на вхід подається, зазвичай, безперервні дані. Тобто мережа може отримувати в якості вхідних даних сигнали, що включають в себе будь-яку кількість показників та можуть мати декілька діапазонів значень. Все залежить від тієї проблеми, що вирішується даною нейронною мережею.

Механізм, для перетворення безперервних вхідних сигналів у двійковий вихід, називається логістичною регресією. Незважаючи на те, що цей алгоритм називається логістична регресія, це насправді алгоритм для вирішення задачі класифікації. І він використовується для класифікації (бінарних чи ні) вхідних даних у відповідні їм класи (додає мітки).

Логістична регресія може використовуватися для різних проблем класифікації, таких як виявлення спаму, прогнозування діабету, чи даний клієнт придбає певний товар, чи буде користувач натискати певне рекламне посилання чи ні, і ще багато інших прикладів. Термін логістика в логістичній регресії використовується, оскільки ми застосовуємо функцію до зваженої суми вхідних даних та параметрів моделі, і саме ця функція називається функцією logit (sigmoid). Ця функція обчислює ймовірність того, що набір входів відповідає правді.

$$F(x) = \frac{1}{1+e^{-x}} \quad (7)$$

Вхідні дані в даному випадку подаються як показник експоненти в знаменнику, адже таким чином наш результат завжди буде додатнім. Це є необхідною умовою, адже отримані безперервні входи повинні бути представлені як певні ймовірності, а ймовірності не можуть бути від’ємними. І,

оскільки ймовірність теж не може бути більшою за одиницю, ми застосовуємо ділення одиниці на необхідні вхідні дані, обмежуючи таким чином наш результат від 0 до 1.

Якщо вхід після проходження функції активації повертає нам значення наближене до 1, то маємо певну впевненість, що етикетка застосовується для подальшого обрахунку. Тоді як значення, близькі до 0, вказують на непотрібність даного входу.

2.12 Аналіз різних мов програмування

Python - це популярна, на даний момент, мова програмування загального призначення, яка має багато сфер застосування. Вона також є однією з найбільш популярних мов для машинного навчання. Це спричинено багатьма факторами: простота, швидкість, велика кількість розробників та готових бібліотек, та ін.

Одним з найважливих пунктів для машинного навчання є дані, і Python має багато бібліотек для роботи з даними, представлених в любих форматах. Ця мова програмування також поєдную всі важливі поняття, такі як попередній аналіз даних, їхня обробка, вилучення об'єктів, пошук аномалій в даних, візуалізація та кластеризація.

Також Python має багато бібліотек для машинного навчання, як і традиційного, так і глибинного. З їхньою допомогою можна легко будувати складні моделі для різних цілей, підганяти їх під себе, міняти параметри та покращувати фінальний результат. Таким чином на даній мові програмування легко реалізувати проекти з різноманітними функціональним можливостями, наприклад класифікація зображень, виявлення спаму, онлайн-прогнозування продаж, прогноз ціни акцій та інші важливі і популярні задачі машинного навчання.

Вважається, що Python найкраще підходить для різних AI проектів . Дана мова, з його простотою, великим співтовариством і численними бібліотеками, дозволяє розробникам легко створювати бажані архітектури, а потім так само легко впроваджувати їх в виробництво.

Також часто використовується така мова програмування як **R**.

Вона створювався спеціально для аналізу даних: запис конструкцій мови зрозуміла багатьом фахівцям в області. В неї є багато функцій, необхідних для аналізу даних, які являються вбудованими функціями мови. Таким чином, перевірка статистичних гіпотез, наприклад, займає лише декілька рядків коду. Ця мова дасть вам широкий статистичний аналіз, як для роботи з даними з любого пристрою, так і для аналізу фінансової чи продуктової моделі.

Також тут зручний репозиторій пакетів і велика кількість готових тестів практично під всі методи машинного навчання. Особливо треба виділити кілька

якісних пакетів для візуалізації даних для різних завдань

Порівняно з Python, R вважається повільним і дещо відсталим, коли мова йде про масштабні продукти з великими наборами даних. При створенні великих проектів краще використовувати Python з його гнучкістю, для реальної розробки продукту.

2.13 Чому Python найпопулярніший для машинного навчання

Звичайний проект штучного інтелекту відрізняється від традиційного програмного проекту. Відмінності полягають у стеці технологій, необхідності глибоких досліджень, необхідних навичках. Щоб реалізувати власні прагнення до машинного навчання, потрібно використовувати таку мову програмування, яка буде стабільною, гнучкою і матиме доступні та популярні інструменти. В мові програмування Python все це можна знайти, тому на даний момент вона є найпопулярнішою для штучного інтелекту.

Python є привабливим, адже він пропонує стислий, простий та читабельний код, завдяки чому його легко вивчити та почати з ним працювати. Навіть попри складні алгоритми машинного навчання та різноманітні процеси, простота Python дозволяє розробникам спокійно писати надійні системи.

Багато різних програмістів вважають Python більш інтуїтивним та зрозумілим, в порівнянні з іншими мовами програмування. Деякі вказують на велику кількість бібліотек і розширень, які в свою чергу спрощують реалізацію різних функціональних можливостей. Загальноприйнято, що Python підходить

для реалізації продукту кількома розробниками, оскільки він може швидко виконувати набір складних завдань машинного навчання і дозволяє легко створювати прототипи, для тестування продукту.

Під час розробки, задля економії часу та енергії, програмісти використовують готові рішення, реалізовані в бібліотеках та фреймворках. Бібліотека ПЗ – це попередньо написаний та оптимізований код, який

розробники можуть використовувати для вирішення своїх задач та проектів, задля збереження часу. Python, має дуже багатий та різноманітний набір технологій та бібліотек для штучного інтелекту і машинного навчання. Ось деякі, найбільш популярні, з них:

- Pandas для заагального аналізу даних
- Seaborn, Matplotlib для візуалізації даних
- NumPy для високопродуктивних обчислювань і аналізу даних
- Scikit-learn для традиційного машинного навчання
- Keras, TensorFlow, і PyTorch для нейронних мереж

Scikit-learn має добре реалізовані та оптимізовані різні популярні алгоритми класифікації, регресії та кластеризації, такі як: векторні машини, випадкові ліси, градієнтні підсилення, к найближчих сусідів, що робить цю бібліотеку однією з найпопулярніших, та найбільш широко використовуваною для машинного навчання в Python [11].

Може виникнути питання коли використовувати яку бібліотеку. Нижче наведена таблиця з частими задачами ШІ та технологіями в Python , які найкраще будуть підходити для вирішення (у таблиці 2.1).

Таблиця 2.1

Data analysis and visualization	NumPy, SciPy, Pandas, Seaborn, Matplotlib
Machine learning	TensorFlow, Keras, Scikit-learn
Machine learning	OpenCV
Natural language processing	NLTK, spaCy

З такими популярними рішеннями, можна просто розвивати потрібний продукт набагато швидше та якісніше. Команді розробників не потрібно заново винаходити велосипед, тому що можна використовувати одну з існуючих бібліотек чи фреймворків для реалізації потрібних функцій. Таких варіант буде навіть швидшим в роботі та більш стабільним. Всього, інтернет-сховища налічують понад 140 тисяч різноманітних програмних пакетів для Python.

Окрім всього Python є кросплатформною мовою програмування, що дозволяє розробникам переносити проект з однієї машини на іншу без будь-яких змін. Python підтримується такими платформами як Linux, Windows і macOS. Код, написаний на цій мові, може також бути використаним для створення виконуваних програм для більшості операційних систем.

Також, розробники використовують послуги хмарних обчислень для своїх потреб, адже так можна запускати свою програму на величезних потужностях, що знаходяться десь на серверах Google чи Amazon. Тоді як навчання моделей машинного навчання на звичайних, повсякденних ноутбуках може зайняти декілька днів. Перенесення коду Python не створює ніяких проблем.

На даний момент Python є швидко зростаючою мовою програмування, що кожного року стає більш затребуваною, як можна бачити на графіку (Рис.8.)

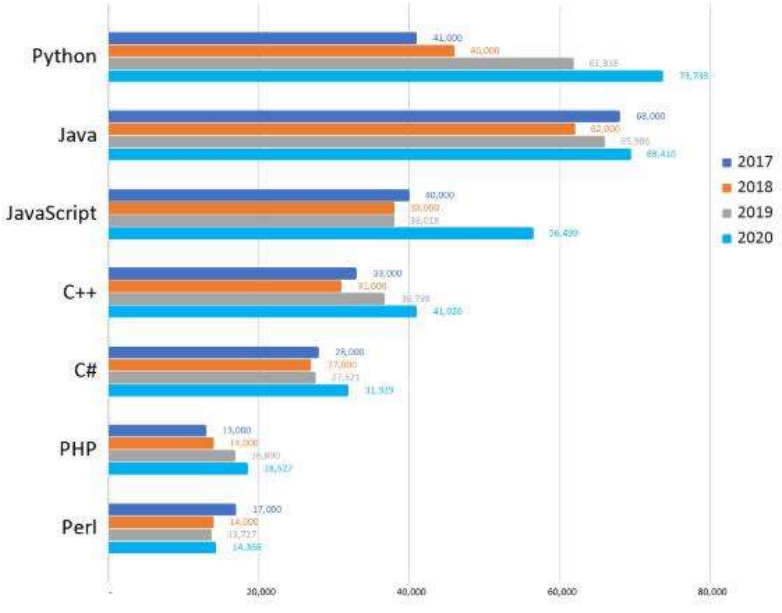


Рис. 2.8 Популярні мови програмування по роках

Окрім того у останніх дослідженнях можна спостерігати, що Python зазвичай використовується для веб-розробки (26%), робота з даними (18%) та машинне навчання(9%). Тобто, якщо додати науку про дані і машинне навчання, вони становлять переможні 27%.[12]

Не менш важливим є факт, що спільнота Python в тематиці машинного навчання постійно росте по всьому світу. Існує багато форумів Python, постійно ведеться активний обмін досвідом, пов'язаним з різними рішеннями для ШІ. В переважній більшості, якщо у вас з’явилось якесь питання чи помилка, ви зможете знайти ще кількох людей з такою ж проблемою, а деколи і з рішенням чи порадою.

2.15 Основні бібліотеки для машинного навчання в Python

Python є найкращим вибором мови для розробників, які хочуть застосовувати аналіз даних, різні статистичні методи або глибинне навчання в своїй роботі.

Однією з найбільших переваг Python для роботи з машинним навчанням є надзвичайно широкий набір різноманітних бібліотек на всі випадки життя.

Бібліотеки – це набори процедур і функцій, підпрограм або об’єктів, написаних на певній мові, які використовуються для розробки програмного забезпечення. Потрібний набір бібліотек може полегшити розробникам виконання різноманітних складних завдань без багаторазового переписування однакових рядків коду, що добре економити час.

Машинне навчання в своїй основі базується на математиці та різних базових алгоритмах. Зокрема, математична оптимізація, статистика і теорія ймовірності. Тому різні «математичні» бібліотеки дуже полегшують життя розробникам в сфері машинного навчання [12].

Scikit-learn - це, мабуть, найкорисніша бібліотека для машинного навчання в Python. Бібліотека sklearn містить безліч ефективних інструментів для машинного навчання та статистичного моделювання, включаючи класифікацію, регресію, кластеризацію та зменшення розмірності.

Scikit-learn поставляється з великою кількістю функцій. Ось декілька з них, які допоможуть вам зрозуміти поширення:

- Практично всі алгоритми «навчання з учителем». Починаючи від узагальнених лінійних моделей (наприклад, лінійна регресія), векторних машин (SVM), дерев рішень до різних баєсових методів - всі вони є частиною панелі інструментів даної бібліотеки. .
- Алгоритми «навчання без учителя». Пропонується велике поширення алгоритмів машинного навчання – таких як кластеризація, факторний аналіз, аналіз основних компонентів.

Всі алгоритми глибинного навчання варто запускати на графічних процесорах комп'ютерів, тому що такий варіант значно пришвидшує час виконання, в порівнянні з запуском на процесорі. Для цього вам не потрібно писати на рівні C ++ або CUDA, можна просто використати відповідні бібліотеки.

TensorFlow – відкрита бібліотека програмного забезпечення для машинного навчання, розроблена Google. Вона використовує власну систему багатошарових вузлів, що дозволяє швидко будувати, навчати та тестувати нейронні мережі для виявлення та розшифровування образів та кореляцій на великими наборами даних на графічних процесорах.

Всі обчислення, зроблені за допомогою TensorFlow виражаються як графи станів потоків даних. Назва TensorFlow походить від операцій, які виконують нейронні мережі над багатовимірними масивами даних. Адже ці багатовимірні масиви називаються «тензорами».

ВИСНОВКИ ДО РОЗДІЛУ 2

У цьому розділі було розглянуто в загальному вигляді сфери застосування, різновиди та популярні алгоритми машинного навчання.

Було розглянуто загальне поняття машинного навчання, глибинного навчання та нейронних мереж, а також різницю між цими термінами. Виділено основні принципові проблеми машинного навчання, такі як: класифікація, кластеризація та прогнозування. На основі цих задач будуються практично всі задачі ШІ. Важливим пунктом є будова нейронної мережі, принципи її роботи та основні пункти її роботи. Оскільки такі мережі на даний момент є основою практично всіх популярних на сьогоднішні день алгоритмів штучного інтелекту.

Окрім цього важливим пунктом є вибір мови програмування, адже куди приємніше працювати з великою кількістю корисних бібліотек, які є необхідними для комфортної розробки програмного забезпечення. Висновком до цього пункту є однозначний вибір мови Python як основної для реалізації всіх задач машинного та глибинного навчання. Також розглянуто основні бібліотеки даної мови програмування, які необхідно освоїти для комфортної роботи з алгоритмами машинного та глибинного навчання.

РОЗДІЛ 3. ОГЛЯД МОДЕЛЕЙ ТА АЛГОРИТМІВ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ

Задачу трекінгу відвідувачів можна умовно розділити на дві складові послідовні задачі. Спочатку, попередньо навчена модель певної архітектури, розпізнає відвідувачів, як звичайних людей на відео. Практично, розпізнавання відбувається на кожному кадрі з відео, беручи його як звичайне зображення. Виходить, що на кожному кадрі відео, модель знаходить нових людей, без розуміння, що це та сама людина з попереднього кадру. Таким чином, однієї моделі є недостатньо для трекінгу відвідувачів, адже вона не бачить зв’язку між однією і тією самою людиною на сусідніх кадрах, що є в свою чергу необхідною складовою даної задачі. Для встановлення такого співвідношення потрібні спеціальні алгоритми, або інші нейронні мережі, навчені під таку задачу.

Спершу, при виявленні об'єкта виділяємо об'єкт у кадрі, ставимо навколо нього обмежувальне поле або маску і класифікуємо знайдений об'єкт. Варто зауважити, що робота детектора закінчується на цьому. Він обробляє кожен кадр відео повністю незалежно та ідентифікує різні об'єкти на кожному конкретному кадрі.

Тепер черга наступного кроку, трекер, повинен відстежувати певний об'єкт протягом всього відео. Якщо детектор виявляє 3 машини у кадрі, об'єктовий трекер повинен ідентифікувати 3 окремі об’єкти і відстежувати їх через наступні кадри (за допомогою унікального ідентифікатора).

3.1 Огляд моделей для розпізнавання людей

Для початку розглянемо різні моделі для розпізнавання людей на зображеннях, адже відео – це всього лиш набір зображень. Загалом, виявлення об'єктів – поширена проблема комп’ютерного зору, яка стосується ідентифікації та розміщення об'єкта певних класів на зображенні. Інтерпретацію локалізації об'єкта можна здійснювати різними способами, включаючи створення

обмежувального поля навколо об'єкта або позначення кожного пікселя на зображенні, яке містить об'єкт (називається сегментацією).

Зазвичай, для виявлення об’єктів використовують згорткові нейронні мережі (CNN) або їхні модифікації, як одні з основних варіантів для розпізнавання та класифікації зображень. Виявлення об'єктів, обличчя розпізнавання тощо – одні з напрямків, де широко використовуються CNN [14].

Обробляючи зображення згорткова нейронна мережа приймає вхідне зображення, обробляє його та класифікує за певними категоріями (наприклад, собака, кішка, тигр, лев). Комп'ютери бачать вхідне зображення як масив пікселів, і його розміри залежать від роздільної здатності зображення.

Є певні популярні архітектури нейронних мереж, які часто використовують як основу для побудови власних моделей для розпізнавання, класифікації та трекінгу об’єктів на різного роду зображеннях.

3.1.1 R-CNN

Ця методика поєднує два основні підходи: застосування згорткових нейронних мереж високої ємності до пропозицій регіону знизу вгору з метою локалізації та сегментації об'єктів; та контролю попередньої підготовки допоміжних завдань.

Після цього відбувається доопрацювання, орієнтоване на домен, що сприяє підвищенню продуктивності. Автори цієї архітектури назвали алгоритм R-CNN (регіони з функціями CNN), оскільки він поєднує пропозиції регіонів з згортковими нейронними мережами.

Ця модель отримує зображення та витягує близько 2000 пропозицій регіону знизу вгору. Потім вона обчислює функції для кожної пропозиції, використовуючи велику CNN. Після цього класифікує кожну область, використовуючи специфічні для класу лінійні підтримуючі векторні машини (SVM).

Система виявлення об'єктів у цій моделі має три модулі. Перший відповідає за створення регіональних пропозицій, незалежних від категорії, які

визначають набір детекторів-кандидатів, доступних моделі. Другий модуль являє собою велику згорткову нейронну мережу, відповідальну за отримання функціонального вектора параметрів фіксованої довжини з кожної області. Третій модуль складається з класу підтримуючих векторних машин (SVM).

Ця модель використовує вибірковий пошук для створення запропонованих регіонів. Вибіркові пошукові групи, схожі за кольором, текстурою, формою та розміром. Ця модель досягає середньої точності в 53,7% на класичних наборах даних.

Проте в архітектури R-CNN є деякі важливі недоліки, які можуть вплинути на вибір моделі, наприклад:

1. Навчання цієї моделі це багатоступеневий конвеєр. Налаштування згорткової нейронної мережі на об'єктних пропозиціях регіонів, пристосування SVM до функцій CNN та нарешті навчання моделі налаштування обмежувального поля.
2. Навчання R-CNN коштує дорого в структурному плані через наявність в складі глибоких мереж, які займають величезну кількість місця в пам’яті.
3. Виявлення об'єктів відбувається повільно, оскільки він виконує перехід CNN для кожної пропозиції об'єкта.

Проте алгоритм є простим і зрозумілим, тому користувався популярністю [15]. Для того щоб позбутись деяких недоліків, на базі цього алгоритму було створено дві популярні модифікації, такі як:

- Fast R-CNN
- Faster R-CNN
- Mask R-CNN

У порівнянні з R-CNN, швидкий R-CNN має певний ряд переваг: він має більш високу середню точність на класичних даних, навчання відбувається на одній стадії тренування, яка оновлює всі шари мережі, а для кешування функцій

не потрібно зберігання всієї моделі.

У своїй архітектурі швидкий R-CNN приймає на вхід моделі як початкове зображення, так і набір об'єктних (регіональних) пропозицій. Потім воно обробляє зображення за допомогою згорткових та максимумом повністю з'єднаних шарів, щоб отримати згорнуту карту особливостей. Потім функціональний вектор фіксованого рівня витягується з кожної карти об'єктів регіоном об'єднання інтересів для кожного запропонованого регіону на зображенні.

Потім вектори функцій подаються на повністю пов'язані шари, після яких вони розгалужуються на два вихідних шари. Один виробляє оцінки ймовірності для декількох класів об'єктів, а інший створює чотири реальних числа для кожного класу об'єктів. Ці 4 числа представляють положення обмежувального поля для кожного з об'єктів: верхню ліву точку, верхню праву, нижню ліву та нижню праву.

Модель швидшого R-CNN складається тільки з двох модулів: глибокої згорткової мережі, відповідальної за пропозицію регіонів, і швидкого детектора R-CNN, який використовує вже запропоновані регіони. Мережа регіональних пропозицій приймає зображення як вхідні дані та генерує вихід прямокутних об'єктних пропозицій. Кожен з прямокутників має оцінку об'єктивності.(Рис. 9)

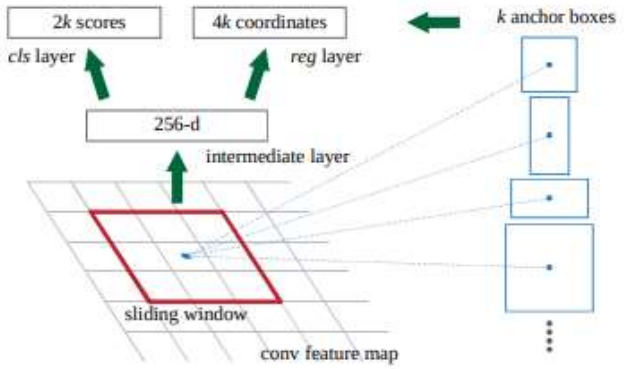


Рис. 3.1. Принцип роботи Faster R-CNN

Ще однією популярною модифікацією алгоритму R-CNN, яка є її розширеною версією, за допомогою якої можна оцінювати людські пози та рухи.

Модель, Mask R-CNN, класифікує та локалізує потрібні об’єкти за допомогою обмежувального поля та семантичної сегментації, яка класифікує кожен піксель на набір категорій. Тобто отриманий результат буде виглядати не як прямокутник, всередині якого знаходиться необхідний об’єкт, а попіксельно виділений шуканий об’єкт. Ця модель розширює швидший R-CNN, додаючи передбачення сегментованих масок для кожної області інтересів. Mask R-CNN в результаті дає два виходи: етикетка класу та обмежувальний набір пікселів [16].

3.1.2 YOLO

Порівняно з іншими мережами класифікації пропозицій регіонів (швидкий чи швидший RCNN), які виконують виявлення, за допомогою різних регіональних пропозицій і, таким чином, виконують передбачення багаторазово для кожного з отриманих регіонів зображення, архітектура YOLO більше нагадує Full CNN (повністю згорнута нейронна мережа) і передає один раз через модель і на виході отримаємо прогноз. Ця архітектура розбиває вхідне зображення на сітку певної розмірності та генерує для кожного сегменту сітки 2 обмежувальних поля та відповідні ймовірності класу для них.

Загалом YOLO розшифровується як You Only Look Once, тобто модель бачить зображення повністю, а не розбитим на регіони. Таким способом виявлення об'єктів стає певною єдиною задачею регресії, від пікселів зображення до обмежувальних координат поля та ймовірностей кінцевого класу

Ця уніфікована модель має ряд переваг перед традиційними методами виявлення об'єктів. По-перше, YOLO надзвичайно швидка модель. Оскільки ми розглядаємо кадр як вхід для задачі регресії, нам не потрібен складний конвеєр. Ми просто запускаємо нашу нейронну мережу на нове зображення в тестовий час, щоб передбачити виявлення.

По-друге, YOLO глобально міркує про результат під час прогнозування. На відміну від методів, що базуються на розсувних вікнах та запропонованих

регіонах, YOLO бачить усе зображення під час тренувань та тестування, тому воно неявно кодує контекстну інформацію про всі об’єкти, а також їх зовнішній вигляд

По-третє, YOLO засвоює узагальнюючі уявлення предметів. Під час навчання природним зображенням та тестування на художніх творах YOLO перевершує найкращі методи виявлення, такі всі модифікації R-CNN з великим запасом.

Дана мережа використовує функції всього зображення для прогнозування кожного обмежувального поля. Вона також передбачає всі обмежувальні поля для всіх класів для зображення одночасно. Це означає, що наша мережа міркує глобально про повне зображення та всі об'єкти на зображенні. Конструкція YOLO дозволяє здійснювати повну підготовку в швидкості реального часу, зберігаючи доволі високу середню точність.

Наша система ділить вхідне зображення на сітку певного розміру, в залежності від розміру вхідного зображення. Якщо центр шуканого об'єкта потрапляє в комірку сітки, ця комірка відповідає за виявлення цього об'єкта (Рис. 10.)

Кожна комірка сітки прогнозує обмежуючі поля та достовірність результату для них відповідно. Ці показники достовірності відображають наскільки впевненою є модель у тому, що поле містить шуканий предмет, а також наскільки точним вона вважає предмет, що воно прогнозує.

Кожне обмежувальне поле складається з 5 прогнозів: x, y, w, h та впевненості. Координати (x, y) являють собою центр вікна відносно меж комірки сітки. Ширина та висота прогнозуються відносно всього зображення. Нарешті, прогноз достовірності являє собою перетином між передбачуваним полем і будь-яким полем даної істини.

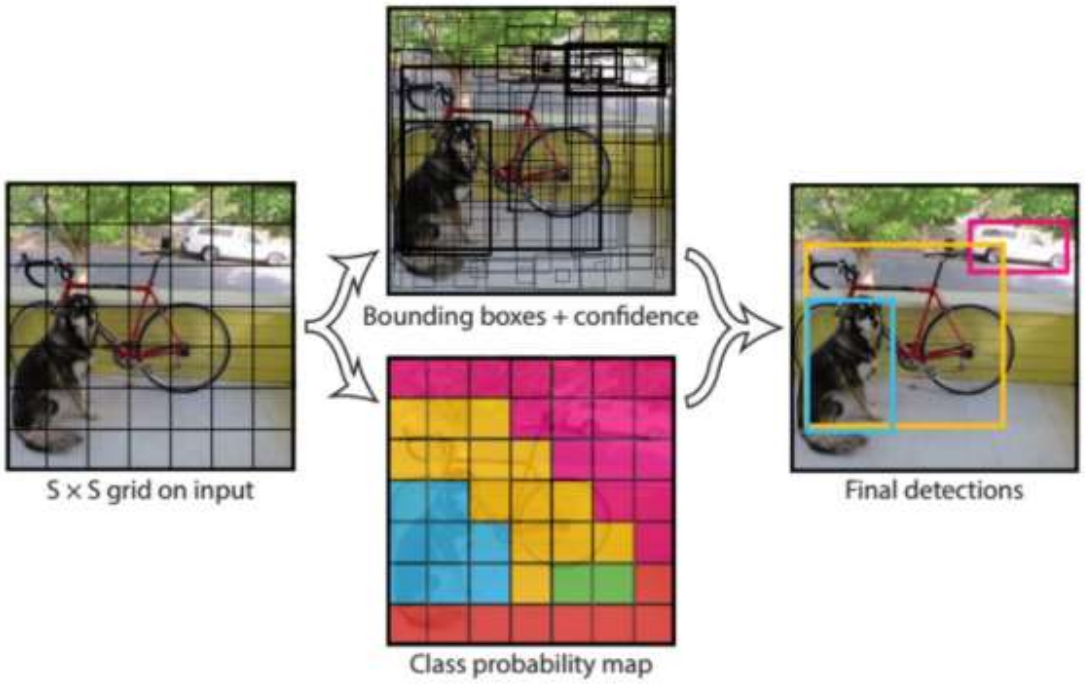


Рис.3.2. Принцип роботи архітектури YOLO

YOLO накладає сильні просторові обмеження для прогнозування обмежувального поля, оскільки кожна комірка сітки передбачає лише два поля і може мати лише один клас. Це просторове обмеження обмежує кількість об'єктів поблизу, які може передбачити наша модель. Наша модель важко справляється з дрібними предметами, які являються групами, наприклад, зграї птахів. Оскільки наша модель вчиться прогнозувати обмежувальні поля з даних, вона намагається узагальнити об'єкти в нових або незвичайних співвідношеннях сторін або конфігураціях. Наша модель також використовує відносно грубі функції для прогнозування обмежувальних полів, оскільки в даній архітектурі є декілька шарів зменшення розміру з вхідного зображення. Таким чином, головне джерело помилок такої архітектури це неправильні локалізації [17].

3.1.3 SSD

Ще однією популярною архітектурою є SSD (Single Shot Detector - детектор одного пострілу, кроку), що призначена для виявлення об'єктів у режимі реального часу. Як вже було сказано, швидкий R-CNN використовує

мережу пропозицій регіону для створення граничних полів і використовує їх для класифікації об'єктів. Хоча такий підхід вважається найкращим у точності, весь процес працює зі швидкістю близько 10 кадрів в секунду при середній якості характеристик машини. Це значно нижче, ніж потребує обробка в режимі реального часу, яка часто є необхідною умовою. SSD прискорює процес, усуваючи потребу в мережі регіональних пропозицій. Щоб відновити отримане падіння точності, SSD застосовує кілька покращень, включаючи багатомасштабні функції та поля за замовчуванням. Ці вдосконалення дозволяють SSD відповідати точності швидшого R-CNN, використовуючи зображення нижчої роздільної здатності, що ще більше підвищує швидкість їхньої обробки. Згідно з наступним порівнянням, він досягає швидкості обробки в режимі реального часу і навіть перемагає точність швидшого R-CNN в деяких випадках. (Точність вимірюється як середня точність прогнозів, отриманих на класичних даних)

Виявлення об'єктів SSD складається з 2 частин:

- Витягувати карти особливостей зображення
- Застосовування фільтрів згортки для виявлення об'єктів

Стандартно, метод виявлення об'єктів у зображеннях представляється за допомогою єдиної глибокої нейронної мережі. Даний підхід під назвою SSD дискретизує вихідний простір обмеження полів у набір полів за замовчуванням, за допомогою різних співвідношень сторін та масштабів, а також за місцем розташування на карті функції. На час прогнозування мережа генерує оцінку за наявність кожної категорії об'єктів у кожному полі за замовчуванням і виробляє коригування для поля щоб краще відповідати формі об’єкта. Крім того, мережа поєднує передбачення з декількох карт особливостей з різною роздільною здатністю, щоб природно обробляти предмети різного розміру. SSD це відносно простий метод, що виявляють розташування об’єктів, оскільки це повністю виключає генерування регіональних пропозицій та їх подальшу перевірку. Цей підхід також виключає функції перекомпонування етапів та інкапсулює всі

обчислення в одній мережі. Це робить SSD простим у навчанні та в інтеграції в системи, які потребують компонента виявлення [18].

Стандартні набори даних. на яких тестуються точності алгоритмів підтверджують, що SSD має конкурентну точність, на рівні з методами, які використовують додатковий крок пропозиції об'єкта, при тому значно швидше відбувається забезпечення єдиної основи для навчання.

3.2 Огляд алгоритмів для трекінгу

Одна з найбільш затребуваних тем в сучасному світі машинного навчання та нейронних мереж це різні способи відстеження (трекінгу) об’єктів. Відстеження об'єкта може бути визначено як процес зосередження на рухомому об'єкті з відео і здатність визначати, чи цей об’єкт присутній у попередньому або наступному кадрах.

Процес відстеження об'єктів можна розбити на три етапи:

1. На вхід приймається початковий набір даних, такі як набір з координат об’єктів, що потрібно відстежувати (результат роботи моделі для виявлення об’єктів)
2. Створення унікального ідентифікатора для кожного з вхідних виявлених об’єктів.
3. А потім відстежувати кожен із об’єктів, коли вони рухаються по кадрах у відео, підтримуючи присвоєний унікальний ідентифікатор

Крім того, такий спосіб відстеження об'єктів дозволяє отримувати унікальний ідентифікатор для кожного відстежуваного об'єкта, що дає змогу рахувати унікальні об’єкти у відео. Відстеження об'єктів має першорядне значення для створення лічильника вхідного та вихідного потоку.

Ідеальний алгоритм відстеження об'єктів:

1. Фаза виявлення об'єкта за допомогою моделі потрібна лише один раз (тобто коли об'єкт спочатку виявлений)
2. Надзвичайно швидка робота - набагато швидша, ніж запуск самого детектора об'єктів

3. Уміння обробляти такі ситуації, коли відстежений об’єкт “зникає” або переміщується за межі відеокадру
4. Є стійким до оклюзії (ситуації, в якій два або більше об'єкти розташовані практично на одній лінії зору і один з них, розташований ближче до спостерігача, частково або повністю закриває інший об'єкт)
5. Уміє підбирати між собою кадри, які він "втратив"

Це насправді висока планка для будь-якого алгоритму обробки зображень або відео, яку важко досягнути. Проте можна скористатися різноманітними хитрощами, які допоможуть вдосконалити об'єктні трекери.

Але перш ніж приступати до побудови алгоритмів, потрібно розуміти які є варіації всіх методів відстеження об’єктів, наприклад:

1. Чисто виявлення руху. Зазвичай від статичної камери і поширений у системах спостереження. Часто виконується лише на піксельному рівні (через обмеження швидкості).
2. Локалізація об'єкта, тобто фокусу уваги на області зображення, що цікавить. Такий спосіб дозволяє зменшити кількість даних для обробки. Часто виявляються лише точки інтересів, які згодом використовуються для вирішення проблеми відстеження та спостереження.
3. Сегментація руху: зображення сегментуються на області, що відповідає різним рухомим об'єктам.
4. Тривимірна форма від руху, яка також називається структура з руху. Таким чином вирішують задачі способами, схожими до стерео зору.
5. Власне саме відстеження об'єктів. Певний набір функцій, за допомогою яких відстежуються контрольні, наприклад кутові, точки

Щоб краще розуміти теорію, для прикладу можна взяти три автомобілі, що в даний момент рухаються. Задача полягає в тому, щоб знайти автомобіль,

який першим дістається до фінішу.

Для цього потрібно зробити декілька речей:

1. Однозначно ідентифікувати автомобіль за якоюсь унікальною цифрою чи буквою
2. Зберігати це число або букву для кожного автомобіля протягом усієї гонки, стежачи за ними під час руху.
3. Автомобіль, який дістається першим, можна визначити за заданим йому унікальним ідентифікатором.

Наведені вище кроки передбачають використання алгоритмів відстеження об'єктів. Обов’язковою умовою для роботи є встановлення детектора об'єктів для виявлення бажаних об'єктів, для подальшого трекінгу.

Це можна зробити за допомогою декількох методів, таких як:

- Використання попереднього кадру
- Використання n-попередніх кадрів

Можна зберігати минулі позиції кожного об’єкта в структурованому форматі і використовувати його в якості основи для визначення того, який унікальний ідентифікатор пов’язаний з яким об’єктом. Рішення працювало б, але не було б масштабованим, оскільки потрібно багато пам’яті для зберігання даних з попереднього кадру, і кожного разу потрібно обробляти n кадрів даних для визначення ідентифікатора [19].

Для того щоб розуміти, як правильно знаходити інформацію про унікальні ідентифікатори, потрібно розглянути декілька популярних алгоритмів трекінгу

3.2.1 Центроїди

Центроїда а даному випадку – це звичайний геометричний центр об’єкта або обмежувального поля навколо об’єкта. Може бути використаним як узагальнення об’єкта на зображенні.

Відстеження на основі центроїд – це зрозумілий, але високоефективний алгоритм відстеження. Існують також більш вдосконалені алгоритми

відстеження на основі ядра та кореляції, проте в своїй основі вони всі працюють по однаковій суті.

Цей алгоритм відстеження об'єктів називається «центроїдним» відстеженням, оскільки він спирається на евклідову відстань між існуючими центрами об'єктів, тобто тими які, вже є ідентифікованими, та новими центроїдами об'єктів між наступними кадрами відео.

Загалом, даний алгоритм відстеження на основі центроїд – це багатоетапний процес.

Першим етапом є прийняття обмежувальних координат рамки та обчислення центроїди. Даний алгоритм відстеження передбачає, що в наборі обмежувальної рамки є координати для кожного виявленого об'єкта в кожному кадрі, за допомогою яких і вираховуються центроїди.

Ці обмежувальні рамки можуть бути виготовлені будь-яким типом детектора об’єктів, за умови, що вони обчислюються для кожного кадру в відео.

Після того, як ми матимемо координатні рамки, ми повинні обчислити “центроїду”, або, простіше кажучи, центр обмежувального поля. Кожен визначений центр по суті являє собою унікальний об’єкт, який потрібно буде відслідковувати, тому кожній обчисленій центроїді призначається ідентифікатор.

Наступним кроком йде обчислення евклідової відстані між новими обмежувальними рамками, виявленими детектором в наступному кадрі та існуючими об'єктами, а точніше центроїдами нових об’єктів та існуючих.

Для кожного наступного кадру в відео, застосовуємо перший крок обчислення центроїди об'єкта; однак, замість того, щоб призначити новий унікальний ідентифікатор кожному виявленому об'єкту (що би перешкоджало меті відстеження об'єкта), спершу потрібно визначити, чи можливо пов’язати новий центроїд об'єкта з одним із існуючи об’єктів. Для виконання цього процесу ми обчислюємо евклідову відстань між кожною парою існуючих центроїд об’єктів та вхідними центроїдами об'єкта з нового кадру.

Але як використовувати евклідові відстані між цими точками, щоб насправді їх відповідно пов'язувати? Відповідь – у наступному кроці.

Третім кроком даного алгоритму виступає оновлення відповідних координат існуючих об'єктів, а також оновлення їхніх центроїд, разом з збереженням унікальних ідентифікаторів.

Основне припущення алгоритму відстеження за допомогою центроїд полягає в тому, що даний об'єкт потенційно може переміщатися між сусідніми кадрами, але відстань між центроїдами для цих кадрів буде меншою, ніж усі інші відстані між об'єктами Тому, якщо вирішити порівнювати центроїди за допомогою мінімальних відстаней між сусідніми кадрами, можна побудувати успішний трекер об’єктів на відео.

Проте, що робити, якщо з’явився новий об’єкт на кадрі, проте відстань між його центром та іншими об’єктами є завеликою для узагальнення?

Наступним кроком у даному алгоритмі йде реєстрація нових об’єктів на кадрах. У випадку, якщо вхідних виявлених об’єктів виходить більше, ніж існуючих, які відслідковуються, потрібно зареєструвати новий об’єкт. "Реєстрація" просто означає, що потрібно додати новий об'єкт до нашого списку відстежуваних об'єктів, що в свою чергу означає:

- Призначення йому нового унікального ідентифікатора
- Збереження центроїда координат обмежувального поля для цього об’єкта для подальших обрахунків

Потім ми можемо повернутися до другого пункту і повторити дану послідовність кроків для кожного кадру в нашому відео.

Не менш важливим є наступний пункт, який включає в себе дереєстрацію старих об’єктів. Оскільки, будь-який розумний алгоритм відстеження об'єктів на відео повинен мати можливість обробляти таку ситуацію, коли об’єкт був загублений, тобто вийшов за межі кадру чи поля зору. В такі моменти потрібно видаляти з пам’яті центроїду даного об’єкту, для оптимізації роботи алгоритму.

Зазвичай використовується така послідовність дій – коли центр об’єкту не можна поєднати ні з яким новим центром, протягом наступних n кадрів, тоді цей об’єкт рахується як той, що вже вийшов з кадру і більше нема суті зберігати інформацію про нього в пам’яті (виняток – алгоритми-лічильники, яким потрібна інформація про всі унікальні об’єкти у відео).

В даного алгоритму відстеження об’єктів є певні обмеження та недоліки, такі як:

- По-перше, даний алгоритм вимагає спрацювання моделі для виявлення об’єктів на кожному кадрі відео, що значно сповільнює його роботу.
- Другий недолік пов'язаний з основними припущеннями самого алгоритму відстеження центроїд – центроїди повинні лежати близько між наступними кадрами, таким чином унеможливлюється швидке пересування об’єктів по кадру, а також часто трапляється явище оклюзії.

Це припущення, як правило, справедливо, але потрібно мати на увазі, що при представленні нашого тривимірного світу за допомогою двовимірних кадрів часто відбувається перекривання одних об’єктів іншими. В такому випадку з точки зору трекера об’єктів може відбуватись перемикання ідентифікатора об’єкта. Якщо два або більше об'єктів перетинаються один з одним при тому, їхні центроїди теж перетинаються, і мають мінімальну відстань до іншого відповідного об'єкта, алгоритм може випадково підміняти ідентифікатор об'єкта. Важливо розуміти, що проблема об'єкта, що перекривається чи закривається, не є характерна тільки для алгоритму відстеження за допомогою центроїд – це часто трапляється і для багатьох інших трекерних систем, у тому числі і для просунутих, та складних.

Однак проблема є найбільш вираженою при алгоритмі центроїд, оскільки ми чітко покладаємось на евклідові відстані між центроїдами та відсутність додаткових метрик, евристики чи вивчених зразків [20].

Та не зважаючи на ці моменти, даний алгоритм є дуже популярним по причині своєї простоти та швидкості, що є дуже важливими показниками в системах, що потребують роботи в реальному часі.

3.2.2 Sort

Найпопулярніший і також доволі простий алгоритмів відстеження - це SORT (Simple Online and Realtime Tracking – простий онлайн-трекер реального часу) . Він може відслідковувати декілька об'єктів у режимі реального часу, адже алгоритм просто пов'язує вже виявлені об’єкти через різні кадри на основі координат результатів виявлення,

SORT – це реалізація візуальних рамок відстеження декількох об'єктів на основі рудиментарної асоціації даних та методів оцінки стану. Він призначений для онлайн-додатків відстеження, де доступні лише минулі та поточні кадри, в такому випадку метод створює ідентичності об'єктів на льоту.

Хоча цей мінімалістичний трекер не вирішує проблему оклюзії чи повторного введення об'єктів, його мета – слугувати базовою лінією та бути стандартним тестом для розвитку майбутніх систем.

Ідея полягає у використанні якоїсь нестандартної моделі для виявлення об'єктів, а потім підключити результати до алгоритму SORT, який відповідає виявленим об'єктам через кадри.

Такий підхід, очевидно, дає багатоцільовий алгоритм: SORT не потрібно знати, який тип об'єкта ми відстежуємо. Навіть нічого не потрібно вивчати: для виконання асоціацій SORT використовує математичну евристику, таку як максимізація метрики перетину правди з отриманим результатом, між обмежуючими рамками в сусідніх кадрах. Кожне поле позначено цифрою, тобто певним ідентифікатором об'єкта, і якщо в наступному кадрі немає відповідної рамки, алгоритм передбачає, що об'єкт залишив кадр.

Якість такого підходу, природно, багато в чому залежить від якості виявлення основного об'єкта. Вся суть оригінальної задумки SORT полягала у тому, щоб показати, що алгоритми виявлення об'єктів настільки вдосконалені,

що вам не потрібно робити щось надто фантазійне щодо відстеження, і ви можете досягти найсучасніших результатів за допомогою прямої евристики. З того часу з'явилися вдосконалення, зокрема, наступного покоління алгоритму SORT, Deep SORT (SORT вийшов у 2016 році, а Deep SORT вже у 2017 році). Він був розроблений спеціально для зменшення кількості перемикань між особами, що накладаються, забезпечуючи стабільність відстеження [21].

3.2.3 Deep Sort

Розглянемо детальніше популярну модифікацію алгоритму SORT, яку називають найпопулярнішою і однією з найбільш широко використовуваних, елегантних систем відстеження об'єктів – Deep SORT.

Deep SORT – це порівняно новий алгоритм відстеження, який розширює попередній алгоритм SORT, і показав чудові результати в проблемі відстеження кількох об'єктів на відео.

Найбільш поширеною є задача одночасного відстеження багатьох об’єктів, при яких кожен кадр має більше одного предмета для відстеження.

Загальний метод вирішення цього питання має два етапи:

- Виявлення
- Встановлення відповідності

Виявлення: спочатку всі об'єкти виявляються у кадрі. Може бути один знайдений об’єкт або декілька. Після того, як ми виявили об’єкт в кадрі, проводиться відповідність для аналогічних виявлень у попереднього кадру. Сусідні кадри дотримуються послідовності для отримання відстеження об'єкта. У Deep SORT цей загальний метод розділений на три етапи.

По-перше, для виявлення об’єктів на зображенні використовується популярний метод виявлення об'єктів на основі CNN або його швидші модифікації. Цей метод є двоступеневим алгоритмом виявленням об'єктів, який добре справляється з виявленням та рухом об'єктів, навіть у випадках перетворення об'єктів та оклюзій.

Проміжний крок до об'єднання даних складається з моделі оцінки. При цьому використовується стан кожного об’єкта як вектор з восьми величин, тобто центр рамки (x, y), шкала рамки, співвідношення сторін рамки та їх похідні по часу як швидкості. Певний фільтр Калмана використовується для моделювання цих станів як динамічної системи. Якщо немає даний об'єкт не виявляється на кількох наступних кадрах, він вважається поза кадром або втраченим. Для нововиявленого вікна запускається нова оцінка.

На останньому кроці, враховуючи передбачувані стани фільтрації Калмана з використанням попередньої інформації та нещодавно виявленого вікна в поточному кадрі, робиться асоціація для нового виявлення зі старими об'єктними треками в попередньому кадрі. Це обчислюється за допомогою певного алгоритму для двостороннього зіставлення графіків. Це робить алгоритм ще більш надійним, встановлюючи ваги відповідності з формулюванням на відстані.

Це добре пояснено на наступній схемі (Рис. 11). Трекер використовує вектор станів для зберігання історичної інформації про попередні виявлення об’єктів. Якщо з'явиться новий кадр, ми можемо попередньо використовувати виявлення обмежувальних рамок або обчислити їх за допомогою методів виявлення об'єктів. Таким чином, при використані даного алгоритму є необов’язковим пунктом використовувати детектор об’єктів на кожному кадрі.

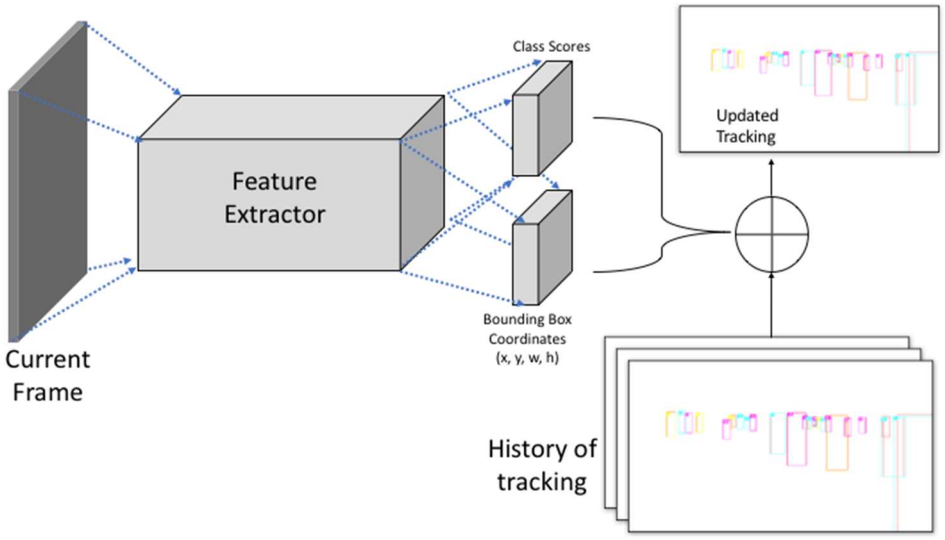


Рис. 3.3. Принципова схема роботи алгоритму Deep SORT [22]

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було розглянуто поняття моделей для розпізнавання шуканих об’єктів на зображеннях, основні задачі як вирішуються за допомогою таких моделей. Нейронні мережі для такого типу задач є одними з найбільш популярних серед різноманітних моделей глибинного навчання на сьогоднішній момент.

Також було розглянуто основні популярні архітектури для вирішення цієї задачі. Наприклад: R-CNN, Fast R-CNN, Faster R-CNN, YOLO та SSD. Кожна з цих моделей має свої переваги та недоліки, які є описаними в даному розділі.

Також було проведено аналіз різних алгоритмів трекінгу, без яких потрібна система не змогла б працювати правильно. Вони є необхідними для встановлення відповідностей між виявленими об’єктами на сусідніх кадрах для того щоб саме відстежувати їх, а не просто знаходити. Такими розглянутими алгоритмами є: алгоритм на основі центроїд, SORT, та його модифікація з добавленням глибинного навчання Deep SORT.

Всі названі архітектури моделей та алгоритми є популярними та широко використовуються в сферах машинного навчання, оскільки є велика кількість задач, під які підходить саме та архітектура чи саме той алгоритм.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В загальному варіанті при розробці будь-якого програмного забезпечення, що працює на основі моделей машинного навчання, можна виділити певні стандартні кроки, такі як:

- підготовка даних для навчання та збирання датасету
- вибір комфортної мови програмування
- вибір підходящих бібліотек машинного навчання для поставленої задачі
- продумування конкретної архітектури моделі
- проведення навчання моделі
- тестування результатів моделі

При розробці системи трекінгу відвідувачів, потрібно враховувати момент, що окрім самої моделі машинного навчання, використовується власне алгоритм відстеження, що ускладнює процес розробки.

4.1 Підготовка датасету

Серед всіх пунктів розробки програмного забезпечення збір та обробка даних є найбільш затратним по часу. Це також є активною темою досліджень у популярних спільнотах машинного навчання та науки про дані. Загалом є дві причини, чому це настільки популярне питання на даних момент. По-перше, машинне навчання в сучасному світі стає все більш популярним та широко використовуваним, знаходяться нові сфери застосування, які на даний момент не мають достатньої кількості позначених даних. По-друге, глибокі методи навчання, такі як нейронні мережі, автоматично генерують особливості для навчання, і через це вимагають дуже великих обсягів розмічених даних.

Збір даних дозволяє зафіксовувати історичні події, щоб потім реально було використовувати аналіз даних та машинне навчання для пошуку повторюваних ситуацій. Завдяки таким даним можна створювати та

застосовувати моделі прогнозування, які є затребуваними у багатьох сферах. Вони використовують алгоритми машинного навчання для пошуку тенденцій, сезонності та передбачення майбутній змін завдяки цим даним.

Попередня обробка даних відіграє важливу роль у машинному навчанні. Припустимо, вам дають зображення будь-якого об'єкта, який ви намагаєтеся вивчити, і всі зображення вирівняні вертикально. Коли після фази навчання ви передаєте зображення того самого об'єкта під кутом, є ймовірність, що програма не зможе розпізнати цей об’єкт. Також можна навчити модель на об'єктах певного розміру, але під час тестування надати зображення з об'єктом або меншого, або більшого розміру, що можу викликати таку ж помилку. Це лише деякі з багатьох проблем, з якими можна зіткнутися з точки зору даних для навчання. Більш офіційним терміном, який використовується для опису таких артефактів, називається створення об'єкта інваріантним для перекладу, обертання та масштабу. Це означає, що незалежно від того, де знаходиться об’єкт у зображенні, при якій орієнтації та наскільки маленький чи великий об’єкт, або вхідне зображення, програма зможе правильно обробити його та визначити об’єкт.

Для роботи нашої системи потрібна модель розпізнавання об’єктів, навчена для визначення локації людей на зображеннях. Для того щоб навчити таку модель потрібно знайти або створити датасет з великою кількістю фото людей в різних позах в різних ситуаціях. Чим більше буде різноманітних варіацій фото в датасеті для навчання – тим точнішою буде модель в результаті.

Одним із популярних відкритих наборів даних, який включає в себе масив зображень з людьми, є COCO датасет. Окрім людей, за допомогою цього датасету можна навчити модель для розпізнавання такий об’єктів як: багато видів тварин, види транспорту, предмети побуту, техніки та багато іншого. Загалом, цей набір даних включає в себе 123,287 зображень на який виділено і позначено 886,284 об’єктів. Сам датасет зберігається у форматі .xml файлів, де позначені координати виявлених об’єктів та їхні ідентифікатори і назви [23].

4.2 Вибір мови програмування та бібліотек

Для розробки даної системи основною мовою розробки вибрано Python версії 3.6., оскільки на даному етапі розвитку машинного навчання найбільш розвинутою та популярною серед мов програмування є саме Python. Цей вибір був детально пояснений та обґрунтований в пункті 2.12.

Для розробки архітектури даної моделі машинного навчання була вибрана популярна бібліотека Keras, що працює поверх надпотужної бібліотеки Tensorflow, що є розробленою компанією Google. Keras є більш зрозумілою бібліотекою, а також дуже гнучкою, з дуже обширним набором функцій для побудови необхідної архітектури для певної задачі. Також дана бібліотека має можливість роботи з GPU, що значно пришвидшує час навчання моделі. Також, таким способом, можна отримати доступ до більш розширеної аналітики для оцінки моделі.

4.3 Навчання моделі для розпізнавання

Навчання потрібної моделі відбувалось на COCO датасеті, у 50 ітерацій, тобто епох. За допомогою тестів, було визначено що після 50 епохи точність перестала суттєво змінюватись і таким чином, подальше навчання не має сенсу. Адже воно привело б тільки що перенавчання мережі на початкових даних, зменшуючи її гнучкість перед тестовими даними. Також, на даному етапі навчання, функція втрат перестала спадати, що показує про максимум отриманих особливостей з цього датасету за допомогою такої архітектури.

Сама архітектура моделі складається з вхідного шару, на який подається зображення для обробки, набору згорткових шарів з малим вікном згортки і з різною глибиною та кількох, останніх, повністю поєднаних шарів. (Рис. 12)

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 8194
=================================================================
Total params: 134,268,738
Trainable params: 126,633,474
Non-trainable params: 7,635,264
_________________________________________________________________
```

Рис. 4.1. Архітектура моделі

Така архітектура, через наявністю повністю з’єднаних шарів дуже довго навчається і результуючий файл з моделлю є великим. Проте в результаті отримуємо хорошу точність. При наступних конфігураціях, що були встановлені на робочі машині, повне навчання моделі, тобто всі 50 епох, зайняли 8 годин. Наявні конфігурації:

- Intel i7-6500U

- Nvidia GTX920
- 12 GB RAM
- 240 Gb SSD

Після навчання моделі на тренувальних даних, було проведено тестування на раніше невідомих мережі зображеннях. Порівняння спрацювання моделі та отриманих точностей показує про достатньо натреновану модель. (Рис.13). результатах навчання моделі, було отримано точність розпізнавання людей на зображенні 96.3%.

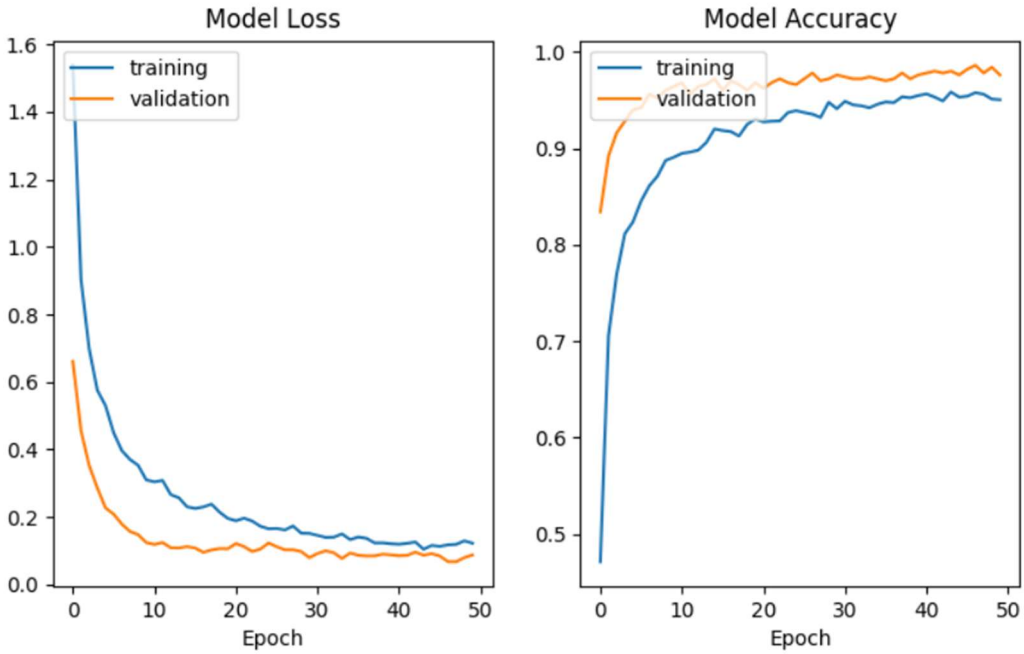


Рис. 4.2. Результати навчання моделі

Отримана модель може розпізнавати та виявляти на зображені людей з швидкістю обробки одного файлу приблизно 0.3 мс. Як результат роботи повертається набір з чотирьох координат (правий нижній кут, правий верхній, лівий нижній та лівий верхній), а також відповідних назв класів (в нашому випадку клас тільки один – людина). За допомогою отриманих координат, на вхідному зображенні можна намалювати обмежувальну рамку навколо шуканих об’єктів для візуалізації результатів роботи нашої моделі. Приклад такого зображення можна побачити на рис.13.

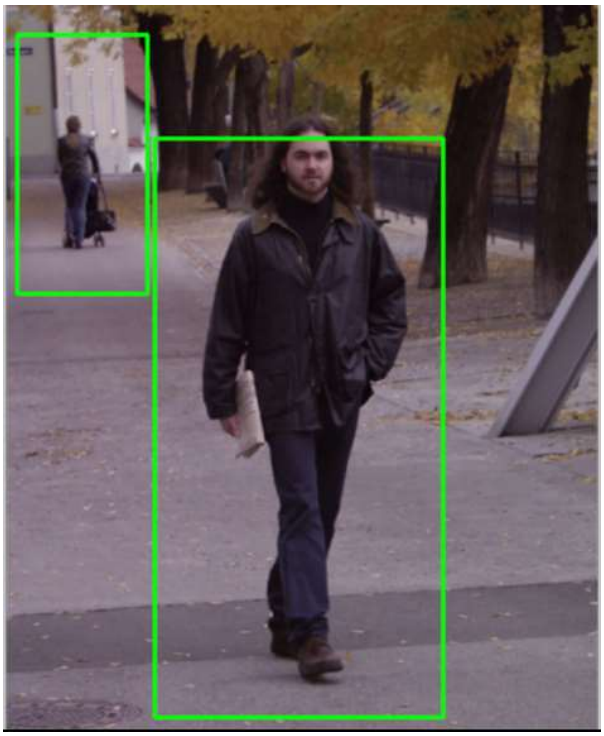


Рис.4.3. Приклад роботи моделі розпізнавання людей

Як можна бачити з наведених вище оцінок точності та з прикладу роботи моделі, дана мережа показує хороший результат та велику ймовірність правильного розпізнавання.

4.4 Алгоритм відстеження людей

Для розробки даної системи було реалізовано алгоритм трекінгу на основі центроїд, так як ставилось за мету, розробити просту і інтуїтивно зрозумілу систему, яка окрім цього, ще й повинна працювати в режимі реального часу. Під такі вимоги ідеально підходить цей алгоритм. Детально він описаний у пункті 3.2.1.

Одною з модифікацій, застосованою у даній системі є те, що модель, а потім і алгоритм обробляють не кожний кадр з відео, а з певною частотою, в даному випадку, кожний 5 кадр. Таким чином можна суттєво пришвидшити роботу спрацювання, і система може працювати в режимі реального часу навіть на не дуже потужному обладнанні. Частоту можна міняти, проте потрібно пам’ятати про баланс між результуючою точністю та швидкістю обробки. Адже при великому вікні, пропускаючи багато проміжних кадрів, можна «губити»

об’єкти, не знаходячи відповідності між кадрами (наприклад, якщо об’єкти рухаються дуже швидко). А при малій частоті, або при обробці всіх кадрів, на комп’ютері з слабкими конфігураціями робота системи буде далекою від режиму реального часу.

Також, в даній системі було реалізовано лічильник загального, вхідного та вихідного потоків відвідувачів для можливості отримання додаткової важливої статистичної інформації. Ця функція була реалізована як модифікація до алгоритму центроїд.

Для початку, ведеться простий підрахунок кількості людей, що загалом з’являються на відео для подальшої обробки інформації. Це відбувається за допомогою сумування всіх унікальних ідентифікаторів, що присвоюються об’єктам алгоритмом протягом всього відео.

Для того щоб реалізовувати лічильних вхідного та вихідного потоків відвідувачів потрібно вибрати «лінію входу» на кадрі. Це може бути певний рівень на відео (певне число в пікселях), який можна вважати за лінію входу, або «двері» (Рис. 14). Це число вибирається в залежності від розташування камери спостереження для кожного випадку окремо. В даному варіанті на тестовому відео, камера знаходиться над входом і приблизно на 300 пікселі знаходиться лінія входу (на зображенні широка зелена лінія).

Тоді при відслідковуванні людей алгоритмом центроїд, відбувається також перевірка на розташування об’єкта в певному проміжку над чи під лінією (на зображенні тонкі зелені лінії) І коли об’єкт, тобто його центр, знаходиться в цій зоні, алгоритм перевіряє в яку сторону він буде рухатись на наступних кадрах – вгору чи вниз по осі У. Якщо пересікає лінію зверху вниз, то значить він заходить в приміщення, а якщо знизу вверх – виходить.

Рис. 4.4. Приклад роботи системи

Щодо отриманої точності роботи системи, то вона рівна 79% для загального лічильника людей. Ця точність рахується як відношення отриманої кількості людей за допомогою моделі і алгоритму до реальної кількості прохожих людей.

$$Accuracy_{total} = \frac{N_{model}}{N_{real}} \quad (8)$$

І 76% при підрахунку вхідного та вихідного потоку, ця точність розраховувалась за допомогою цієї ж формули.

Система працює а режимі реального часу, що означає, що обробка одного відео займає стільки ж часу, скільки і триває це відео. Це зроблено з тією метою, щоб в майбутньому модернізувати систему для обробки потокового відео напряму з камери спостереження.

ВИСНОВКИ ДО РОЗДІЛУ 4

В даному розділі було розгялянуто всі кроки розробки системи трекінгу відвідувачів за допомогою нейроної мережі на основі відео з камер спостереження. При роботі над системою було враховано всі стандартні кроки розробки програмного забезпечення за допомогою машинного навчання.

Було проведено пошук підходящого датасету для навчання моделі розпізнавання людей. Основними вимогами є велика кількість зображень людей різного типажу в різному одязі та в різних проекціях. В результаті вибрано датасет COCO, що знаходиться в вільному доступі як найбільший набір потрібних даних. Його аналіз, а також переваги та зміст теж надані в даному розділі

Було також проведено аналіз різних моделей виявлення людей, в результаті якого було Вибрано архітектуру моделі на базі R-CNN, так як вона є найшвидшою при оптимальних показниках точності. Було проведено тестування різних алгоритмів трекінгу для вибору оптимального для поставленої задачі і в результаті вибрано алгоритм трекінгу на базі центроїд, як найбільш зрозумілий та швидкий.

Також описано принцип роботи лічильника вхідного та вихідного потоку відвідувачів, який був використаний у даній системі, на основі результатів роботи алгоритму трекінгу.

ВИСНОВКИ

Представлений дипломний проект присвячений розробці системи трекінгу відвідувачів на основі відео з камери спостереження, що знаходиться біля входу в приміщення за допомогою алгоритмів машинного навчання.

Наявність такої система дозволяє любому виду приміщення, без різниці комерційне воно чи державне, отримувати інформації про його безпеку в режимі реального часу, дозволяє економити ресурси та дає можливість постійно отримувати економічно важливі статистичні дані. Такі показними є надзвичайно важливими, адже отримується інформація про всі елементи вашого бізнесу, яку можна використати для підвищення ефективності організації та керування.

Така система будується за допомогою двох елементів: нейронної мережі для виявлення відвідувачів на кадрі, а також алгоритму для трекінгу, який буде поєднувати кадри між собою. Перш за все необхідно було вибрати підходящу модель для виявлення людей.

Загалом існує багато видів архітектури подібних моделей, кожна з яких має свої плюси і мінуси. Оскільки дана робота зосереджена саме над забезпеченням можливості роботи системи в реальному часі, то було вибрано доволі просту архітектуру моделі, яка зможе швидко обробляти зображення.

Вибір алгоритму трекінгу теж відбувався за схожою схемою – потрібно забезпечити максимальну швидкість роботи. Таким чином вибрано реалізовувати алгоритм на онові центроїд. Окрім швидкості, цей алгоритм є дуже простим у розумінні, тому його вибір буде легко обгрунтувати зацікавленому клієнту.

Необхідним пунктом для роботи моделі є велика кількість даних для її навчання. Окрім того, що їх повинно було багато, вони повинні бути якісними та різноманітними. В даній роботі було вибрано датасет COCO, як основне джерело зображень для тренування моделі., адже в ньому є надзвичайно велика кількість розмічених зображень людей різного формату.

Загалом, дана розроблена система може легко бути застосована для будь-якого бізнесу, адже є простою в розумінні та корисною в своїй основі. Вона може використовуватися як для великих підприємств і потужних організацій, так і для дрібних стартапів чи магазинів. Оскільки основною мовою програмування для написання даної системи було вибрано Python, це робить її зрозумілою та багатоплатформенною. Тобто дану систему можна запускати на будь-якій машині з любою операційною системою.

Кожна система повинна мати певне майбутнє, і в даному випадку в разі необхідності додаткового функціоналу, дану систему можна легко розширювати завдяки її простоті та модульності. Наступними кроками для покращення системи може бути подальше тестування різних моделей для виявлення, різних алгоритмів трекінгу, а також написання повноцінного додатку для керування системою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кріс Сміт Конверсія: як перетворити ліди в продажі [Текст]. – Альпіна Паблішер – 2019. -248с.
2. Technologies of tracking people [Електронний ресурс]. – Режим доступу до ресурсу: https://behavioranalyticsretail.com/technologies-tracking-people/
3. V-count– офіційна сторінка [Електронний ресурс]. – Режим доступу до ресурсу: https://ru.v-count.com/
4. Sensmax– офіційна сторінка [Електронний ресурс]. – Режим доступу до ресурсу: https://sensmax.eu/solutions/people-counting-software-1/
5. Linkanalytix– офіційна сторінка [Електронний ресурс]. – Режим доступу до ресурсу: https://www.linkanalytix.net/people-counting/
6. When Machine Learning fails [Електронний ресурс] – Режим доступу до ресурсу: https://www.louisdorard.com/blog/when-machine-learning-fails
7. Clustering in ML [Електронний ресурс] – Режим доступу до ресурсу https://www.geeksforgeeks.org/clustering-in-machine-learning/
8. Time Series Prediction [Електронний ресурс] – Режим доступу до ресурсу: https://machinelearningmastery.com/time-series-prediction-with-deep-learning-in-python/
9. Що таке машинне навчання [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/machine-learning-overview.html
10. Класичне машинне навчання. Задачі [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/classical-machine-learning.html
11. Нейронна мережа прямого поширення [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/нейронна_мережа_прямого_поширення

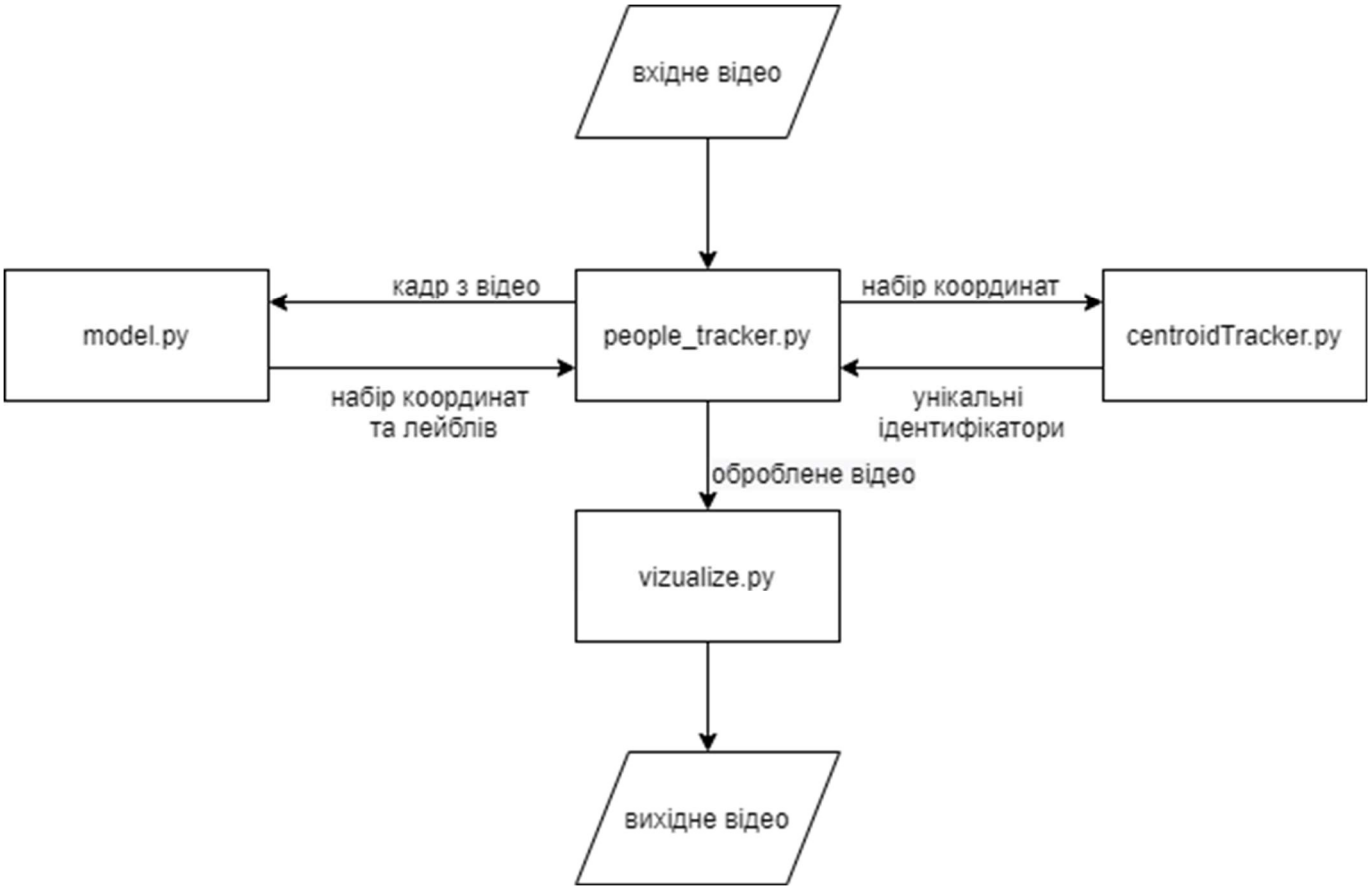
12. Saurabh Kapur Data Preprocessing for Computer Vision [Текст] – O`Reilly – 2017. – 642с.
13. Wes McKinney Python for Data Analysis [Текст] – O`Reilly – 2017. – 544с.
14. Jiang Xiaoyue. Deep Learning in Object Detection and Recognition [Текст] – Springer Nature Singapore Pte Ltd., 2019. — 237 с.
15. Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html
16. Fast R-CNN [Електронний ресурс]. – Режим доступу до ресурсу: https://arxiv.org/abs/1504.08083
17. YOLO [Електронний ресурс]. – Режим доступу до ресурсу: https://arxiv.org/abs/1506.02640v5
18. SSD [Електронний ресурс]. – Режим доступу до ресурсу: https://arxiv.org/abs/1512.02325
19. Multiple Object Tracking Algorithms [Електронний ресурс]. – Режим доступу до ресурсу: https://medium.com/@manivannan_data/multiple-object-tracking-algorithms-a01973272e52
20. Centroid Detection and Tracking Algorithm [Електронний ресурс]. – Режим доступу до ресурсу: https://shodhganga.inflibnet.ac.in/bitstream/10603/91871/13/13_chapter%204.pdf
21. SORT [Електронний ресурс]. – Режим доступу до ресурсу: https://arxiv.org/abs/1602.00763
22. Deep SORT [Електронний ресурс]. – Режим доступу до ресурсу: https://nanonets.com/blog/object-tracking-deepsort/#deep-sort?
23. COCO Dataset – офіційна сторінка [Електронний ресурс]. – Режим доступу до ресурсу: http://cocodataset.org/#home

ДОДАТКИ

ДОДАТОК 1

СТРУКТУРНА СХЕМА

Київ - 2020

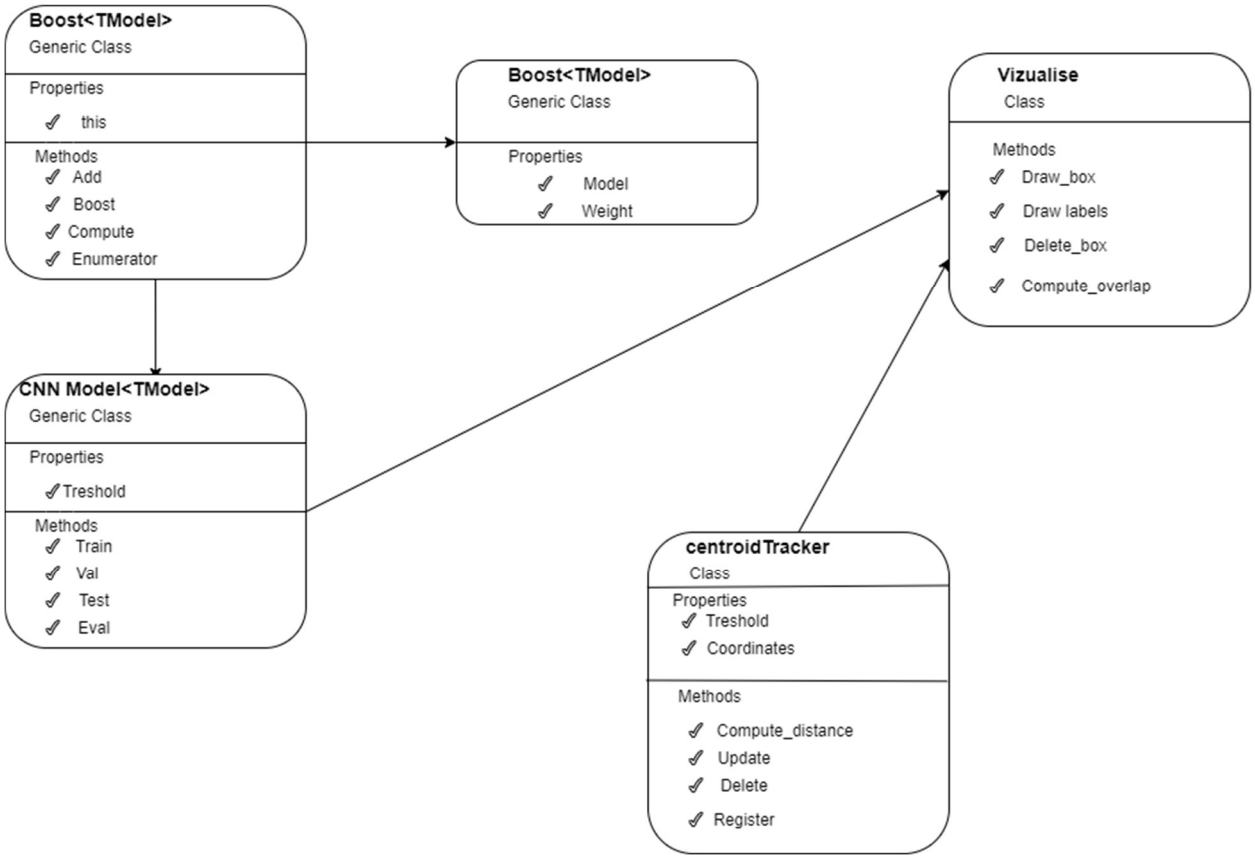


					ДП.4665.05.000 А2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Мірчук Н.П.			Структурна схема	Лит.	Аркуш 1 / Аркушів 1
Перевірив		Волокита А.М.					
Н.Контр.		Сімоненко В.П.				НТУУ «КПІ», ФІОТ, гр. ІО-62	
Затвердив							

ДОДАТОК 2

ФУНКЦІОНАЛЬНА СХЕМА

Київ-2020

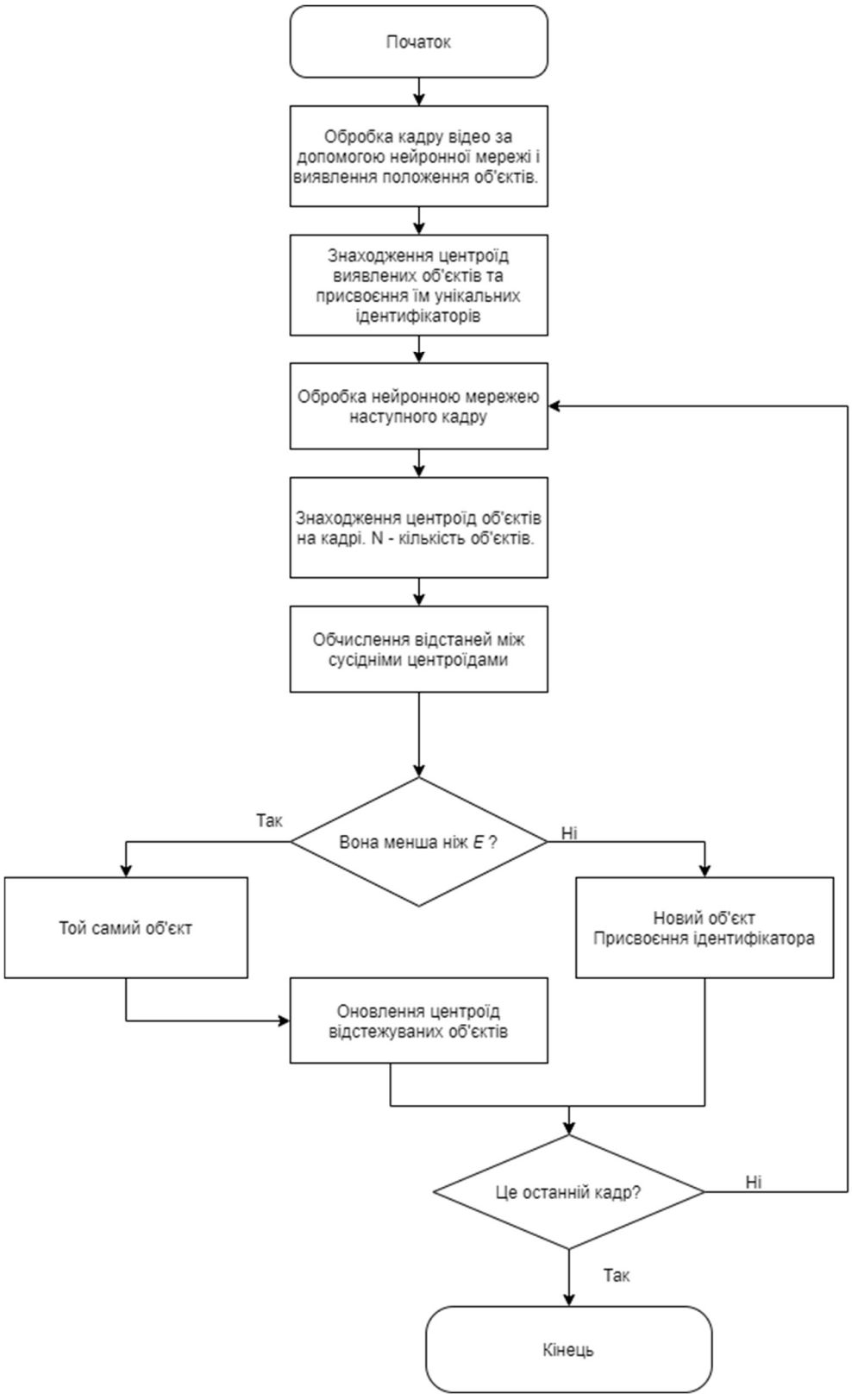


					ДП.4665.05.000 А2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Мірчук Н.П.			Функціональна схема	Лит.	Аркуш Аркушів
Перевірив		Волокита А.М.					1 1
						НТУУ «КПІ», ФІОТ, гр. ІО-62	
Н.Контр.		Сімоненко В.П.					
Затвердив							

ДОДАТОК 3

ПРИНЦИПОВА СХЕМА АЛГОРИТМУ

Київ-2020



					ДП.4665.05.000 А2	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Міpчук Н.П.			Принципова схема алгоритму	Лит. / Арку / Аркуші: 1 / 1
Перевірив		Волокита А.М.				
Н.Контр.		Сімоненко В.П.				НТУУ «КПІ», ФІОТ, гр. ІО-62
Затвердив						

ДОДАТОК 4

ОСНОВНІ ЧАСТИНИ КОДУ ПРОГРАМИ

Київ-2020

centroidTracker.py

```
# import the necessary packages
from scipy.spatial import distance as dist
from collections import OrderedDict
import numpy as np

class CentroidTracker():
    def __init__(self, maxDisappeared=20):

        self.nextObjectID = 0
        self.objects = OrderedDict()
        self.disappeared = OrderedDict()
        self.maxDisappeared = maxDisappeared

    def register(self, centroid):

        self.objects[self.nextObjectID] = centroid
        self.disappeared[self.nextObjectID] = 0
        self.nextObjectID += 1

    def deregister(self, objectID):

        del self.objects[objectID]
        del self.disappeared[objectID]

    def update(self, rects):

        if len(rects) == 0:

            list_of_keys = list(self.disappeared.keys())
            for objectID in list_of_keys:
                self.disappeared[objectID] += 1

                if self.disappeared[objectID] > self.maxDisappeared:
                    self.deregister(objectID)

            return self.objects

        inputCentroids = np.zeros((len(rects), 2), dtype="int")

        for (i, (startX, startY, endX, endY)) in enumerate(rects):
            cX = int((startX + endX) / 2.0)
            cY = int((startY + endY) / 2.0)
            inputCentroids[i] = (cX, cY)
```

```
if len(self.objects) == 0:
    for i in range(0, len(inputCentroids)):
        self.register(inputCentroids[i])


else:
    objectIDs = list(self.objects.keys())
    objectCentroids = list(self.objects.values())

    D = dist.cdist(np.array(objectCentroids), inputCentroids)

    rows = D.min(axis=1).argsort()

    cols = D.argmin(axis=1)[rows]

    usedRows = set()
    usedCols = set()

    for (row, col) in zip(rows, cols):

        if row in usedRows or col in usedCols:
            continue

        objectID = objectIDs[row]
        self.objects[objectID] = inputCentroids[col]
        self.disappeared[objectID] = 0

        usedRows.add(row)
        usedCols.add(col)

    unusedRows = set(range(0, D.shape[0])).difference(usedRows)
    unusedCols = set(range(0, D.shape[1])).difference(usedCols)

    if D.shape[0] >= D.shape[1]:
        for row in unusedRows:
            objectID = objectIDs[row]
            self.disappeared[objectID] += 1

            if self.disappeared[objectID] > self.maxDisappeared:
                self.deregister(objectID)

    else:
        for col in unusedCols:
            self.register(inputCentroids[col])

return self.objects
```

models.py

```
import tensorflow as tf
import keras.backend as K
import keras.layers as KL
import keras.models as KM

class ParallelModel(KM.Model):

    def __init__(self, keras_model, gpu_count):

        self.inner_model = keras_model
        self.gpu_count = gpu_count
        merged_outputs = self.make_parallel()
        super(ParallelModel, self).__init__(inputs=self.inner_model.inputs,
                                            outputs=merged_outputs)

    def __getattribute__(self, attrname):

        if 'load' in attrname or 'save' in attrname:
            return getattr(self.inner_model, attrname)
        return super(ParallelModel, self).__getattribute__(attrname)

    def summary(self, *args, **kwargs):

        super(ParallelModel, self).summary(*args, **kwargs)
        self.inner_model.summary(*args, **kwargs)

    def make_parallel(self):
     ginal model placed on different GPUs.

        input_slices = {name: tf.split(x, self.gpu_count)
                        for name, x in zip(self.inner_model.input_names,
                                           self.inner_model.inputs)}

        output_names = self.inner_model.output_names
        outputs_all = []
        for i in range(len(self.inner_model.outputs)):
            outputs_all.append([])

        for i in range(self.gpu_count):
            with tf.device('/gpu:%d' % i):
                with tf.name_scope('tower_%d' % i):
                    zipped_inputs = zip(self.inner_model.input_names,
                                        self.inner_model.inputs)
```

```
                inputs = [
                    KL.Lambda(lambda s: input_slices[name][i],
                              output_shape=lambda s: (None,) + s[1:])(tensor)
                    for name, tensor in zipped_inputs]
                outputs = self.inner_model(inputs)
                if not isinstance(outputs, list):
                    outputs = [outputs]
                for l, o in enumerate(outputs):
                    outputs_all[l].append(o)

        with tf.device('/cpu:0'):
            merged = []
            for outputs, name in zip(outputs_all, output_names):
                def add_dim(tensor):
                    """Add a dimension to tensors that don't have any."""
                    if K.int_shape(tensor) == ():
                        return KL.Lambda(lambda t: K.reshape(t, [1, 1]))(tensor)
                    return tensor
                outputs = list(map(add_dim, outputs))

                merged.append(KL.Concatenate(axis=0, name=name)(outputs))
        return merged

if __name__ == "__main__":

    import os
    import numpy as np
    import keras.optimizers
    from keras.datasets import mnist
    from keras.preprocessing.image import ImageDataGenerator

    GPU_COUNT = 2

    ROOT_DIR = os.getcwd()

    MODEL_DIR = os.path.join(ROOT_DIR, "logs/parallel")

    def build_model(x_train, num_classes):

        tf.reset_default_graph()

        inputs = KL.Input(shape=x_train.shape[1:], name="input_image")
        x = KL.Conv2D(32, (3, 3), activation='relu', padding="same",
                      name="conv1")(inputs)
        x = KL.Conv2D(64, (3, 3), activation='relu', padding="same",
                      name="conv2")(x)
```

```
    x = KL.MaxPooling2D(pool_size=(2, 2), name="pool1")(x)
    x = KL.Flatten(name="flat1")(x)
    x = KL.Dense(128, activation='relu', name="dense1")(x)
    x = KL.Dense(num_classes, activation='softmax', name="dense2")(x)

    return KM.Model(inputs, x, "digit_classifier_model")

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = np.expand_dims(x_train, -1).astype('float32') / 255
x_test = np.expand_dims(x_test, -1).astype('float32') / 255

print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)

datagen = ImageDataGenerator()
model = build_model(x_train, 10)

model = ParallelModel(model, GPU_COUNT)

optimizer = keras.optimizers.SGD(lr=0.01, momentum=0.9, clipnorm=5.0)

model.compile(loss='sparse_categorical_crossentropy',
          optimizer=optimizer, metrics=['accuracy'])

model.summary()

model.fit_generator(
    datagen.flow(x_train, y_train, batch_size=64),
    steps_per_epoch=50, epochs=10, verbose=1,
    validation_data=(x_test, y_test),
    callbacks=[keras.callbacks.TensorBoard(log_dir=MODEL_DIR,
                                write_graph=True)]
)
```

Visualize.py

```
import random
import itertools
import colorsys
import numpy as np
from skimage.measure import find_contours
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import matplotlib.lines as lines
```

```
from matplotlib.patches import Polygon
import IPython.display

import utils

def display_images(images, titles=None, cols=4, cmap=None, norm=None,
                   interpolation=None):

    titles = titles if titles is not None else [""] * len(images)
    rows = len(images) // cols + 1
    plt.figure(figsize=(14, 14 * rows // cols))
    i = 1
    for image, title in zip(images, titles):
        plt.subplot(rows, cols, i)
        plt.title(title, fontsize=9)
        plt.axis('off')
        plt.imshow(image.astype(np.uint8), cmap=cmap,
                   norm=norm, interpolation=interpolation)
        i += 1
    plt.show()

def random_colors(N, bright=True):

    brightness = 1.0 if bright else 0.7
    hsv = [(i / N, 1, brightness) for i in range(N)]
    colors = list(map(lambda c: colorsys.hsv_to_rgb(*c), hsv))
    random.shuffle(colors)
    return colors

def apply_mask(image, mask, color, alpha=0.5):

    for c in range(3):
        image[:, :, c] = np.where(mask == 1,
                                  image[:, :, c] *
                                  (1 - alpha) + alpha * color[c] * 255,
                                  image[:, :, c])
    return image

def draw_box(image, box, color):
    """Draw 3-pixel width bounding boxes on the given image array.
    color: list of 3 int values for RGB.
    """
    y1, x1, y2, x2 = box
    image[y1:y1 + 2, x1:x2] = color
    image[y2:y2 + 2, x1:x2] = color
    image[y1:y2, x1:x1 + 2] = color
```

```
    image[y1:y2, x2:x2 + 2] = color
    return image
```

people_tracker.py

```
from centroidtracker import CentroidTracker
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import timeit
import os
import sys
import random
import math
import skimage.io
import utils
import coco
import model as modellib
import visualize
import urllib.request as urllib2
import particle
import particleFilter as pf
from imutils.video import WebcamVideoStream
from imutils.video import FPS
from imutils.video import FileVideoStream
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd

def main():
    ct = CentroidTracker()
    (H, W) = (None, None)

    bb = []
    ROOT_DIR = os.getcwd()
    MODEL_DIR = os.path.join(ROOT_DIR, "logs")

    if not os.path.exists(COCO_MODEL_PATH):
        utils.download_trained_weights(COCO_MODEL_PATH)

    IMAGE_DIR = os.path.join(ROOT_DIR, "images")
```

```
    class InferenceConfig(coco.CocoConfig):

        GPU_COUNT = 1
        IMAGES_PER_GPU = 1

    config = InferenceConfig()
    config.display()

    model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR, config=config)

    model.load_weights(COCO_MODEL_PATH, by_name=True)

    class_names = ['BG', 'person', 'bicycle', 'car', 'motorcycle', 'airplane',
                   'bus', 'train', 'truck', 'boat', 'traffic light',
                   'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird',
                   'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear',
                   'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie',
                   'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball',
                   'kite', 'baseball bat', 'baseball glove', 'skateboard',
                   'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup',
                   'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
                   'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',
                   'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed',
                   'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote',
                   'keyboard', 'cell phone', 'microwave', 'oven', 'toaster',
                   'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors',
                   'teddy bear', 'hair drier', 'toothbrush']

    font = cv2.FONT_HERSHEY_SIMPLEX
    bottomLeftCornerOfText = (900, 80)
    fontScale = 0.5
    fontColor = (255, 255, 255)
    lineType = 2

    fvs = FileVideoStream(r'C:\Users\DS\Downloads\Citrus_sample.avi').start() #818   2019 02 0
5 13 33 14_2019 02 05 14 02 36
    #video_capture = WebcamVideoStream(src=r'C:\Users\DS\Downloads\Citrus_sample.avi').sta
rt()

    import timeit
    i = 0
    frame_skipped = 5
    dict_of_id_time = {}
    fps = 25#15
    start_time_global = timeit.default_timer()
    while fvs.more():
```

```
        frame = fvs.read()
        frame = fvs.read()
        frame = fvs.read()
        frame = fvs.read()
        frame = fvs.read()
        frame = imutils.resize(frame, width=1080)

        # fps.update()
        start_time = timeit.default_timer()
        print('st')
        results = model.detect([frame], verbose=0)
        print('end model time',timeit.default_timer() - start_time)
        r = results[0]

        rects = r['rois']
        labels = r['class_ids']
        #print('deb')
        rects = rects[np.where(labels==1)]
        labels = labels[np.where(labels==1)]

        for rect in rects:
            xmin = rect[1]
            ymin = rect[0]
            xmax = rect[3]
            ymax = rect[2]
            rect[0] = xmin
            rect[1] = ymin
            rect[2] = xmax
            rect[3] = ymax

        for bnd, label in zip(rects, labels):
            cv2.rectangle(frame, (int(bnd[0]), int(bnd[1])), ((bnd[2]), int(bnd[3])), 1, 1)
                #print(bnd)

        objects = ct.update(rects)

        for (objectID, centroid) in objects.items():

            if objectID not in dict_of_id_time.keys():
                dict_of_id_time[objectID] = i*frame_skipped
            bb.append(centroid)
            text = "ID {}".format(objectID)
            cv2.putText(frame, text+' time '+str("{0:.1f}".format((i*frame_skipped-
dict_of_id_time[objectID])/fps)), (centroid[0] - 10, centroid[1] - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
            cv2.circle(frame, (centroid[0], centroid[1]), 4, (0, 255, 0), -1)
```

```
        frame = cv2.putText(frame, str('Count of people:' + str(ct.nextObjectID)),
                            bottomLeftCornerOfText,
                            font,
                            fontScale,
                            fontColor,
                            lineType)
        # show the output frame
        print(ct.nextObjectID)
        print('Frame id = ', i)

        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        print('Time elapsed: ', str(timeit.default_timer() - start_time_global))
        i += 1
        if key == ord("q"):# or ct.nextObjectID==102:
            print(ct.nextObjectID)
            break

    # do a bit of cleanup
    data = pd.DataFrame(bb)
    print('Time elapsed: ', str(timeit.default_timer() - start_time_global))
    data.to_csv('Result___.csv')
    print(ct.nextObjectID)
    cv2.destroyAllWindows()
    fvs.stop()

if __name__ == '__main__':
    main()
```